DOCUMENT 1/2
DOCUMENT NUMBER
@: unavailable

1. JP,2002-528797,A
2. JP,2004-508616,A

BACK NEXT

MENU SEARCH
HELP

JAPANESE [JP,2002-528797,A]

CLAIMS DETAILED DESCRIPTION TECHNICAL FIELD PRIOR ART DRAWINGS CORRECTION OR AMENDMENT

[Translation done.]

DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
(Technical field to which an invention belongs)
This invention relates to the system and protocol which support in detail the commercial transaction between the plat forms which have various architecture about the

Drawing selection
Representative drawing

ID=000003

[Translation done.]

DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
(Technical field to which an invention belongs)
This invention relates to the system and protocol which support in detail the commercial transaction between the plat forms which have various architecture about the system and protocol which support the transaction (dealings) between the various clients connected to the network.
[0002]
(PRIOR ART)
The communication network of the Internet or others provides the means for communication between the computer platform and those who are being used for broad various transactions, including the commercial transaction at which a participant deals in goods and service. Many efforts are being made in order to promote the commercial transaction on the Internet. However, since there is a standard with which many compete, in order to perform a transaction. The authorized personnel involved in the transaction have to agree with the protocol used beforehand, and they often need the custom-made integration (integration for exclusive use) of the plat-form architecture in order to support this transaction. The commercial process into the specific node which is not compatible may need repair considerable for integration with other nodes for the standard on which it has agreed. If one company (Company) commits one standard or other standards, it will be fixed to the transaction authorized personnel involved in a standardization group, and the company will eliminate except [its].
[0003]
The outline of the challenge which encountered development of Internet commerce is "Eco System:An Internet Commerce Architecture". Tenenbaum outside work: It is indicated to

Computer and 1997; 48-55 pages of May(s).

[0004]

In order to open the commercial transaction on the Internet, standardization of an architecture framework is desirable. The plat form developed in order to support this commercial framework, Commerce Point;Microsoft of IBM. NCA(Network Computing Architecture); of ONE (Open Network Environment);Oracle of Internet Commerce Frameworks;Netscape. And soft JECF (JAVA Electronic Commerce Framework) of Sun/JAVA is included.

[0005]

. In addition to the framework of these monopolies, are based on CORBA IIOP (Common Object Request Broker Architecture Internet ORBProtocol). Program art like the object model distributed in common is being carried out. Use of the object model of the common distribution is for simplifying carrying out migration (exchange) of the system of a company to the system which can be used cooperatively with the business application level for electronic commerce technology. However, the consumers or business using one framework cannot perform a transaction on a different network. This has restricted growth of an electronic commerce system.

[0006]

Probably the company which carries out one framework has an application program interface (API), and this API differs from API which supports other frameworks. Therefore, it is very difficult for a company to access business service mutually, without needing adoption of a common business system interface. Although development of a business system interface with this API level needs the remarkable cooperation between authorized personnel, it is not made often actually.

[0007]

Therefore, it is desirable to provide the framework which promotes the dialog (interaction) between the various plat forms in a communication network. This system will promote the spontaneous trade between the partners of dealings, without needing prior consent for custom-made integration or the wide standard which comes out industrially. This system will promote the advance to business automation, and will eliminate many of the time and costs of a traditional system integration, and risks.

[0008]

It is desirable to provide generally the electronic commerce system transposed to an open market in the closed commercial partner network based on the standard of monopoly.

[0009]

(Outline of an invention)

This invention provides the infrastructure (base = infrastructure) for connecting business with a customer, a supplier company, or a commercial partner. Under the infrastructure of this

invention, two or more companies exchange information and service mutually using the document (for example, document of an XML (extended markup language) system which can be easily understood among partners) which a self-definition can machine read. The document (in this book, it is called a business interface definition, i.e., BID) which describes the document which should be exchanged is sent on the Internet, or communicates to a network member. A business interface definition (BTD) tells the document for using it, when communicating in the service which a company provides for a potential commercial partner, and its service. Therefore, the customer can perform order by a typical business interface definition by submitting purchasing order (purchasing order according to the document definition published by BID of the party which receives the purchasing order). When a supplier company downloads the stock status report according to the document definition published by BID of business system managed stock data, it becomes possible to check the validity. By use of the business document which was defined beforehand and in which machine reading is possible, the more nearly intuitive flexible method for accessing the application of an ENTA prize (company) is provided.
[0010]
The node within a commercial network the interface for the transaction by this invention which comprises the specification which can machine read an interface, It is established with the data structure including the interpretation information for the specification which can machine read the interface and in which machine reading is possible. The specification which can machine read an interface includes the definition of an input document, and the definition of the output document.
The aforementioned node is accepted by the transaction processor which operates as an interface, and these are made.
The definition of an input document and the definition of an output document include the logic structure for description of the set of a storage unit, and the set of a storage unit where the standard XML system document was followed, respectively, for example. The data structure including the interpretation information according to various viewpoints of this invention and in which machine reading is possible, The specification of the data type for the logic structure in the definition of an input and an output document. The content model (for example, list of possible values) for (for example, a string, an array, etc.), and logic structures, and/. Or the semantic definition (for example, mapping code for generating a name) of logic structure is provided including the data structure which carries out the map of the set which the storage unit for a specific logic structure to each entry of the list of order defined beforehand.
[0011]
According to other viewpoints of this invention, an interface contains the repository (this memorizes the library of logic structure, and the interpretation information on logic structure) in

the memory which can be accessed by at least one network node. This repository may be extended so that the library of a definition of an input and an output document, the library of the specification of an interface, and the library of the specification of the interface node of the participant in network may be included.

[0012]

Therefore, the participant in the transaction framework of this invention performs a transaction between the nodes of network plurality including two or more node execution processes included by the transaction. The method of this invention includes memorizing the specification which can machine read the interface for a transaction, and a specification includes the definition of an input document, and the definition of an output document. The definition of an input and an output document includes description of the set of a storage unit, and the set logic structure of a storage unit, respectively. In a desirable system, those definitions are the methods of following the definition of a standard XML document type, i.e., DTD, and are extended and expressed by semantic mapping, content modeling, and some element data typing. The participant in a transaction receives the data which contains a document through a communication network. A participant analyzes the document according to the specification memorized for the transaction, and identifies the input document for the transaction. At least a part of input document is provided after the analysis of a document in the format in which machine reading is possible to the transaction process of providing an output. Based on the definition of the output document of the memorized specification, an output document is formed so that the output of a transaction process may be constituted. An output document is sent to the place of the others which it was sent on the communication network, and was typically returned to the sauce of the input document, or were doubled with the needs of a transaction specific type.

[0013]

Therefore, the definition of a business interface carries out the bridge of the gap during the program executed on the front end of the transaction processing service in the document and the special node which are specified by XML, for example. these front ends -- a JAVA virtual machine -- or other common architecture given to the interconnection of the system of network both ends is realized. The definition of a business interface provides the art in which a transaction protocol is programmed using the definition document of a business interface. The program over the protocol of this transaction is given by the detailed formal specifications of document form.

[0014]

The machinery reading specifications of the interface of a transaction are made accessible to other plat forms in a network. A participant's plat form contains the resources which design the input document and output document by the transaction interface specified in a complementary node. Therefore, a participant's node contains the resources which access the definition of the output document for the definition of the input document for a complementary interface, and a complementary interface. The specifications stored for the accessed participant node are established by including at least a part of definition of the output document of the complementary interface in the definition of the input document of the interface stored in the specifications.

[0015]

The process of establishing the specifications in which the interface by other features of this invention was stored includes accessing from a repository the element of the specifications in which machinery reading is possible. This repository stores the definition of the document which has the logical structure used in order to build the library and interface description document of an outline map for the logical structure, a content model, and the logical structure. In a network, an accessible repository makes construction of description of an easily sharable interface easy. The input document and the complementary process which were formed to the special process negotiate for all differences between the output documents expected as a return easily by agreeing with the common logical structure expressing the communication on a network, and special information.

[0016]

The specifications which can machinery read the interface of the transaction by one feature of this invention contain the document according to the definition of the interface document shared among network members. The definition of an interface document has the logical structure for storing the reference to the definition of the identifier of a special transaction, at least one of the definitions, and the special input-and-output document for transactions. That is, the interface description document to special service actually includes the definition of an input-and-output document. It can contain the pointer of somewhere in positions in the repository of such a definition, or networks in instead of.

[0017]

According to other alternate methods of this invention, the specifications in which machinery reading is possible, In order to store the identifier of an interface, And the business interface definition BID document according to the definition of an interface document including the logical structure for storing the reference to the specifications of the group of one or more transactions supported by at least one and the interface of specifications is included. A document contains the reference to the definition of the input-and-output document to at least one and the special transaction of a definition to each supported transaction.

[0018]

According to other features of this invention, the storage unit defined as the definition document of the input-and-output document has the analyzed data containing the text sentence character which encodes alphabetic data, and markup data which identifies the group of the storage unit by the logical structure of an input-and-output document. According to other features of this invention, at least one of the groups of a storage unit encodes two or more text sentence characters which give natural language. This makes easy use of a definition of an input-and-output document by these documents' readers and developer of human being.

[0019]

According to other features of this invention, the specifications of an input-and-output document include the translation information over at least one of the groups of the storage unit identified by the logical structure. The translation information in one example encodes the definition document to the analyzed character set. In other examples, translation information is provided for a content model specification. For example, in order to make the description in a catalog, the specific logical structure is required to carry the list of many of codes by which the map was carried out. In a certain system, the storage unit in the logical structure of a document includes the set of the data which is not analyzed to satisfy the necessity for special efficiency.

[0020]

According to other features of this invention, the transaction process in a special plat form has the transaction processing architecture which is one of two or more of the modification transaction processing architecture. Therefore, a participant node contains the resources for changing at least a part of input DOKYOMENTO into the format which can be read at the modification transaction processing architecture using information of a transaction. The element of a document definition contains the variable and method by the modification transaction processing architecture of a transaction process. The special field in a document is not only translated into the JAVA object containing data, but obtains and sets up the function relevant to a JAVA object to the participant who has a JAVA virtual machine which works as a front end of a transaction process. Other transaction processing architecture which can be supported by this invention, Common Object Request. Broker Architecture. CORBA and Component Object. Processing according to Model COM, On-line Transaction Processing OLTR, and the interface description language in the meaning of Electronic Data Interchange EDI is included.

[0021]

According to other features of this invention, the repository which stores the document form used in two or more transactions is provided. The specifications in which machinery reading for a special transaction is possible define at least one of the input-and-output documents by referring to the document form in a repository. According to other features, the document form included in a repository has the document form for identifying the participant in a network.

Since such a document form identifies a participant, and specifies the service supported by the participant and specifies the input-and-output document to each of these services, it provides structure.
[0022]
an above-mentioned method -- in addition, this invention -- a network interface, as mentioned above, The memory for storing the program of the data containing the specifications which can machinery read the interface for a transaction, and a command, And the device which manages the transaction between the nodes containing the data processor connected to the memory and the network interface is provided. The program of the command stored in this device contains the logic which performs the process mentioned above for the participant in a transaction.
[0023]
This invention provides the device for establishing the participant interface for the transaction further performed on the network which performs transaction processing by the transaction processing architecture, and the system containing data-processing resources. This device includes the program of the command stored in the accessible medium by the system which provides the tool which builds the definition of the participant interface for a participant [in / efficiency is possible and / a special transaction] by a system. The definition of a participant interface includes the definition of an input document, and an output document definition. An input-and-output document definition has a description document which can machinery read [each] the set of the storage unit in the logical structure for the set of a storage unit, and it can follow the definition of XML document form as one feature of this invention.
[0024]
The device which sets up the participant interface by this gestalt of this invention, The program of the command memorized by the accessible medium with the data processing system is included, That the command which can be executed by a system in order to change into a corresponding data structure the document into which **** was inputted should be compiled, And in order to change the output of transaction processing into the set of the storage parts store of an output document, and logic structure. In order to compile the data structure corresponding to the logical structure of the input which adapts itself to the transaction processing architecture that the command which can be executed by a system should be compiled, and an output document, and the set of a storage parts store, the definition of an input and an output document is answered.
[0025]
The tool for building the definition of the participant interface in a desirable system, In order to access the component of the definition from a complementary node of the repository which memorizes the library of the interpretation information over the logical structure and the logical

structure which are used in order to build interface description, the command which can be executed by a system is included. According to various gestalten of this invention, a repository includes not only the library of the logical structure but the definition of a document provided with the format for specifying the logical structure and a participant interface. According to this gestalt of this invention, a tool is formed in order to build the specification of the business interface by the art mentioned above in relation to description of a participant node. The tool for compiling the tool and interface for building the interface by this gestalt of this invention to a resource required for communication with the transaction processing architecture, I -- it is used for development and optimization of interface description when network use grows based on the interface description which is carried out by the participant node in a desirable system, and defines an input and an output document.
[0026]
therefore, another gestalt of this invention provides the device containing the data processor which executes the memory and the command which was alike and was memorized by the memory containing the compiler which performs the function which mentioned above tool **** for building the definition of a participant interface.
[0027]
According to another gestalt of this invention, use of participant interface description enables operation of a market maker node. The transaction supported by an interface in such a node, And the method of managing a transaction of providing memorizing the machine-readable specification of two or more participant interfaces which identify each input and output document of such a transaction is provided. As mentioned above, the input and the output document are provided with each description of the set of the logical structure to a storage parts store and a storage parts store so that according to an XML standard. All of the definition of a transaction and the definition of a participant interface are provided with the document specified by the art which adapts itself to XML or other standardization document expression languages. In such a market maker node, data provided with the document is received in a communication network. A document is parsed by specifications, in order to identify an input document in one or more transactions which accept the identified input document. At least a part of input document is supplied in a machine-readable format to transaction processing related with one or more identified transactions. The method of supplying at least a part of input document to transaction processing includes performing routing processing by the processing architecture in a market maker node. Routing processing in one gestalt of this invention is performed by the specific processing architecture, and at least a part of input document is changed into the format of the processing architecture of routing processing. The conversion by a desirable gestalt includes generating the programming object containing the variable and method by the processing architecture of routing processing.

[0028]
According to another gestalt of this invention, a market maker node supports repository structure again. Then, since access by the participant in the network to the repository memorized by the market maker node is permitted, processing is included in a market maker node. As mentioned above, in order that a repository may build a transaction interface document, the definition of the logical structure, Interpretation information, the instance of a business interface definition which identifies a participant including the document definition for participant nodes, and the transaction performed by each participant are included.
[0029]
Conversion of the input document to the variable processing architecture of processing to which a document is sent by various substitution to routing processing, The combination of sending an input document to a remote process node in the first document format with a network (routing) or such processing is included. In substitution, routing processing includes the art for changing again the input document defined by one input document definition into a different document defined with different document specification to the processing registered in order to stand by an input document.
[0030]
A market maker node is provided by this invention as the memory which has memorized the program and data of a command containing the specification of the network interface and the participant interface, and a device containing a data processor. Logic is supplied by the data processor in forms, such as a program of a command, in order to perform market maker processing which was mentioned above.
[0031]
Therefore, this invention provides sharing of the specification of an input and output documentation with the foundation over ****** electronic commerce. A document provides intuitive and the method of being supple for carrying out rather than carrying out program creation of the APIs to access easier business service. It is still easier than an always different business system interface to carry out interconnection of such a company by the document exchanged, if the company already agrees on the whole. In addition, such a document is specified in a human readable (human being can be read) format by the desirable example. According to this invention, a business interface is easily specified by human being and computer by a document like the XML document which can be interpreted.
[0032]
Use of the document base transaction architecture of this invention, Many necessity over the custom program for extracting information from each participant in network, and unifying including use of parsing which operates by the fundamentally same method as all the sauce of a document, is removed. Then, unifying the business information from accounting, purchase,

manufacture, delivery, and other functions can be attained by changing into the document which has the architecture according each sauce to this invention first, and processing the data stream subsequently parsed. Each node of the system which participates in a market only needs to get to know about the format of the document specified by the format of the internal system, and a transaction for exchange. Then, there is no necessity for the native (peculiar) format of all other nodes that wish to participate in an electronic commerce network being generable.

[0033]

For perfect business integration, this invention provides the repository of an XML component, an attribute, or the standardization logical structure like metadata, The server for processing the means for carrying out the map of between the semantic (meaning) language to a specific commerce community and a different metadata description and a document, and calling suitable application and service is set up. The fundamental constructing block of the network by this invention, In order for a company to provide what kind of on-line service for a potential trading business partner and to call service. In order to generate interface definition; which announces which document is used, and a trading community and to combine the set of an inside and external business service mutually, the server which supplies a bridge is included. The server operates in order to parse an input document and to call suitable service. The server by this invention is in the format of each host system, and deals with the conversion task from the format of the document which receives from those formats and transmits. Then, the trading partner needs to agree only with the structure, contents, and sequencing of the exchanged business document, and does not need to agree with the details of an application programmer interface. Action brought about as a result of reception of the method by which a document is processed, and a document is based on the business which completely provides service. This raises integration on a business level from a system level. It can make the interface to the partner which was clear and was stabilized show business in spite of the change in the internal technical operation, an organization, or processing.

[0034]

All the processings which can build a business interface definition and can make a server manage commerce by such description, A company, service and a business description primitive like a product, a catalog, It is easily carried out by the common business library of the information model to the general business concept include the standard measurement including a classification of a purchase order, business form like an invoice, time and a date, a place, and goods, or the repository.

[0035]

He could understand other gestalten of this invention by taking into consideration the drawing explained below, detailed explanation, and a claim.

[0036]
(DETAILED DESCRIPTION)
This invention is explained in detail in connection with a drawing. Drawing 1 explains the support of an exchange of the input specified according to the previous statement of the network of a market maker and a market participant, and this interface based on use of a business interface definition, and an output document. This network contains two or more nodes 11-18 mutually connected via the telecommunication or the data communication network of communication networks, such as the Internet 19, or others. Each of the nodes 11-19 comprises computers, such as a portable computer, a desktop personal computer, a workstation, a network of a system, or other data-processing resources. A memory for this node to memorize the business interface definition, In order to support the processor for performing the transaction process of supporting a commercial transaction transaction with other nodes in a network, and such service, the computer program executed by this processor is included. Each node includes the network interface for providing the communication to which the Internet 19 or other communication networks are crossed.
[0037]
In the environment of drawing 1, the nodes 11, 12, 13, 14, 16, and 18 are the specified market participants. The market participant contains the resource about the consumer or supplier of the goods dealt with according to the commercial transaction transaction established according to this invention, or service.
In this example, the nodes 15 and 17 are market maker nodes. The market maker node contains the resource for registering the business interface definition called BID registry. The participant can send a document to a market maker, this document is identified there, and it is shipped by the suitable participant registered so that such a document may be received as an input. A market maker promotes a commercial transaction network by maintaining the repository of the standard form which constitutes the common business library for using it in construction of a business interface definition again.
[0038]
In this example, direct continuation of the market participant 18 is carried out to the market maker 17 via the Internet 19. The direct continuation to this market maker Public networks, such as the Internet 19, And point two point connection as shown between the private connection or the nodes 17 and 18 like a Local Area Network by [actual] introducing, It is shown that the gestalt of the network which supports a commercial transaction transaction will be able to become very various. The actual communication network is quite various and suitable to use it in order to establish the commercial transaction transaction network by this invention.
[0039]

Drawing 2 is a heuristic diagram of the rounded structure in the business interface definition (BID) established for the market participant in the network by this invention. The business interface definition shown in drawing 2 is a data structure which consists of the storage unit and the logical structure which were arranged according to the formal definition of document structures, such as XML document type definition DTD. The structure of drawing 2 includes the 1st logical structure 200 for identifying Bertie. The transaction 204 for the logical structure 201, the physical address 202 and the network address into which it marched for conveying a name or the position 203, and service of a lot is related with the logical structure 200. About each transaction in service sets, the interface definition containing transaction BID205, transaction BID206, and transaction BID207 is provided. In each transaction BID, such as transaction BID205, The logical structure for the name 208, the position 209 on the network with which service is performed, the operation 210 performed by service, and the input document 211 shown with the tag of a lot to be included is provided. Service BID205 contains the output document 212 shown with the tag of a lot. The input document 211 of the lot includes the business interface definition about each input document to which it pointed so that service might answer, including the input document business interface definitions 213, 214, and 215. The business interface definition about an input document contains the module 218 which should be conveyed in a document as it was shown by the name 216, the position 217 on the network which can find out description of a document, and the field. Similarly, the output document set 212 includes the interface definition about the output document containing output document BID219, output document BID220, and output document BID221. About each output document BID, the position 223 of a name 222 and network top or other places and the module 224 of the document are specified. The business interface definition about a participant shown in drawing 2 contains the pointer or other references to the position which can find out the actual definition of the logical structure which should be used about the input and output document of each service, or a definition.

[0040]

In the desirable system, although the document of drawing 2 is specified in XML document type definition DTD, it includes the translation information about the logical structure which can also use other document definition architecture and is used for translation of a request of a document. Each of the transaction BID, the input document BID, and the output document BID is specified according to an XML document type definition. An XML type document is an example of a system based on the analyzed data containing markup data and character data. Markup data identifies the logical structure in a document, and the group of character data identifies the contents of the logical structure. The data which is not analyzed may have inside of a document conveyed for the various purpose. For example, In WWW.W3.ORG/TR/1998/REC-XML-19980210. Refer to the specifications of Extensible Mark-

up Language XML 1.0 REC-XML-19980210 published by the WC3 XML workgroup.
[0041]
In this way, in the illustrated system, the participant node in a network founds a virtual company by carrying out interconnection of a business system and the service to the document which business receives and is generated and by which XML coding was carried out. For example, the specific business interface definition of service establishes returning the document corresponding to BTD of catalog entry, when the document corresponding to BID of the demand to catalog entry is received. When it can accept for the terminal which received the document corresponding to BID of a purchase order, and it received, the document corresponding to BID of a bill is returned. The node in a network processes an XML document, before an XML document goes into the local business system established according to various transaction processing architecture of the arbitrary predetermined systems in a network. In this way, a system is MIME. - Groups of a related document, such as a group etc. of the coded XML document, are decoded, this is analyzed, and the stream of a "markup message" is created. This message uses the following event listeners models, for example, and is shipped by a suitable use and service.
[0042]
The document exchanged by a business service compartment is encoded using the XML language (common business language) generated from the repository of the building block by which a more complicated document definition is generated, and in which it deals. A business description primitive [like a company, service, and a product] whose repository is, A catalog, a purchase order, a business form like an invoice, time, The module of the interpretation information which focused on the function and information common to a business domain containing what gives a day, the measured value of a standard like a position, a classification code, and the interpretation information on the logical structure of an XML document is memorized.
[0043]
Furthermore a business interface definition plays a role of a schema used in order to design the interface for which the document according to this invention is exchanged, it is a document of a high level. Then, a business interface buries the gap between the document described according to XML, and the program executed by the front end of transaction processing service in a specific node. Such a front end is carried out by other common architecture which specifies internal connection of the system in a JAVA virtual machine or a network. Then, a business interface definition provides the art in which it is used for programming a transaction protocol using a business interface definition document. The program for the protocol of a transaction is prescribed by the detailed formal specification of the document type.
[0044]

The business interface definition BID of an example based on the document of the market authorized personnel according to an XML format is given to below. Market authorized-personnel DTD relates a contact and address information with the description of service and fiscal information, and classifies the business information about market authorized personnel. This business information comprises the pointer to exclusive classification for a name, a code, an address, and a business organization to be shown and the term of business. In addition, the authorized personnel answer and the service identified by market authorized-personnel DTD indicates the input and output document to generate. Then, the document which specifies the schema which uses the common business language of an example for market authorized-personnel DTD described by XML with notes which is useful for explanation, service DTD, and transaction document DTD follows below.

[0045]

Market Participant Sample

```
<!DOCTYPE SCHEMA SYSTEM "bidl.dtd">
<SCHEMA>
<H1>Market Participant Sample BID</H1>
<META
WHO.OWNS="Veo Systems"       WHO.COPYRIGHT="Veo Systems"
WHEN.COPYRIGHT="1998"        DESCRIPTION="Sample BID"
WHO.CREATED="*"              WHEN.CREATED="*"
WHAT.VERSION="*"             WHO.MODIFIED="*"
WHEN.MODIFIED="*"            WHEN.EFFECTIVE="*"
WHEN.EXPIRES="*"             WHO.EFFECTIVE="*"
WHO.EXPIRES="*">
</META>

<PROLOG>
<XMLDECL STANDALONE="no"></XMLDECL>
<DOCTYPE NAME="market.participant">
<SYSTEM>markpart.dtd</SYSTEM></DOCTYPE>
</PROLOG>

<DTD NAME="markpart.dtd">
<H2>Market Participant</H2>
<H3>Market Participant</H3>
<ELEMENTTYPE NAME="market.participant">
<EXPLAIN><TITLE>A Market Participant</TITLE>
<SYNOPSIS>A business or person and its service interfaces.</SYNOPSIS>
<P>A market participant is a document definition that is created to describe a business and at
least one person with an email address, and it presents a set of pointers to service interfaces
located on the network. In this example, the pointers have been resolved and the complete BID
is presented here.</P></EXPLAIN>
<MODEL><CHOICE>
<ELEMENT NAME="business"></ELEMENT>
<ELEMENT NAME="person"></ELEMENT>
</CHOICE></MODEL></ELEMENTTYPE>
```

```
<H3>Party Prototype</H3>
<PROTOTYPE NAME="party">
<EXPLAIN><TITLE>The Party Prototype</TITLE></EXPLAIN>
<MODEL><SEQUENCE>
<ELEMENT NAME="party.name" OCCURS="+"></ELEMENT>
<ELEMENT NAME="address.set"></ELEMENT>
</SEQUENCE></MODEL>
</PROTOTYPE>

<H3>Party Types</H3>
<ELEMENTTYPE NAME="business">
<EXPLAIN><TITLE>A Business</TITLE>
<SYNOPSIS>A business (party) with a business number attribute.</SYNOPSIS>
<P>This element inherits the content model of the party prototype and adds a business number
attribute, which serves as a key for database lookup. The business number may be used as a
cross-reference to/from customer id, credit limits, contacts lists, etc.</P></EXPLAIN>
<EXTENDS HREF="party">
<ATTDEF NAME="business.number"><REQUIRED></REQUIRED></ATTDEF>
</EXTENDS>
</ELEMENTTYPE>

<H3>Person Name</H3>
<ELEMENTTYPE NAME="person">
<EXPLAIN><TITLE>A Person</TITLE></EXPLAIN>
<EXTENDS HREF="party">
<ATTDEF NAME="SSN"><IMPLIED></IMPLIED></ATTDEF>
</EXTENDS>
</ELEMENTTYPE>

<H3>Party Name</H3>
<ELEMENTTYPE NAME="party.name">
<EXPLAIN><TITLE>A Party's Name</TITLE>
<SYNOPSIS>A party's name in a string of character.</SYNOPSIS></EXPLAIN>
<MODEL><STRING></STRING></MODEL>
</ELEMENTTYPE>
```

```
<H3>Address Set</H3>
<ELEMENTTYPE NAME="address.set">
<MODEL><SEQUENCE>
<ELEMENT NAME="address.physical"></ELEMENT>
<ELEMENT NAME="telephone" OCCURS="*"></ELEMENT>
<ELEMENT NAME="fax" OCCURS="*"></ELEMENT>
<ELEMENT NAME="email" OCCURS="*"></ELEMENT>
<ELEMENT NAME="internet" OCCURS="*"></ELEMENT>
</SEQUENCE></MODEL>
</ELEMENTTYPE>

<H3>Physical Address</H3>
<ELEMENTTYPE NAME="address.physical">
<EXPLAIN><TITLE>Physical Address</TITLE>
<SYNOPSIS>The street address, city, state, and zip code.</SYNOPSIS></EXPLAIN>
<MODEL><SEQUENCE>
<ELEMENT NAME="street"></ELEMENT>
<ELEMENT NAME="city"></ELEMENT>
<ELEMENT NAME="state"></ELEMENT>
<ELEMENT NAME="postcode" OCCURS="?"></ELEMENT>
<ELEMENT NAME="country"></ELEMENT>
</SEQUENCE></MODEL>
</ELEMENTTYPE>

<H3>Street</H3>
<ELEMENTTYPE NAME="street">
<EXPLAIN><TITLE>Street Address</TITLE>
<SYNOPSIS>Street or postal address.</SYNOPSIS></EXPLAIN>
<MODEL><STRING></STRING></MODEL>
</ELEMENTTYPE>

<H3>City</H3>
<ELEMENTTYPE NAME="city">
<EXPLAIN><TITLE>City Name or Code</TITLE>
```

```
<P>The city name or code is a string that contains sufficient information to identify a city within
a designated state.</P>
</EXPLAIN>
<MODEL><STRING></STRING></MODEL>
</ELEMENTTYPE>

<H3>State</H3>
<ELEMENTTYPE NAME="state">
<EXPLAIN><TITLE>State, Province or Prefecture Name or Code</TITLE>
<P>The state name or code contains sufficient information to identify a state within a designated
country.</P></EXPLAIN>
<MODEL><STRING DATATYPE="COUNTRY.US.SUBENTITY"></STRING></MODEL>
</ELEMENTTYPE>

<H3>Postal Code</H3>
<ELEMENTTYPE NAME="postcode">
<EXPLAIN><TITLE>Postal Code</TITLE>
<P>A postal code is an alphanumeric code, designated by an appropriate postal authority, that is
used to identify a location or region within the jurisdiction of that postal authority. Postal
authorities include designated national postal authorities.</P></EXPLAIN>
<MODEL><STRING DATATYPE="string"></STRING></MODEL>
</ELEMENTTYPE>

<H3>Country</H3>
<ELEMENTTYPE NAME="country">
<EXPLAIN><TITLE>Country Code</TITLE>
<P>A country code is a two-letter code, designated by ISO, that is used to uniquely identify a
country.</P></EXPLAIN>
<MODEL><STRING DATATYPE="country"></STRING></MODEL>
</ELEMENTTYPE>

<H3>Network Addresses</H3>
<ELEMENTTYPE NAME="telephone">
<EXPLAIN><TITLE>Telephone Number</TITLE>
```

```
<P>A telephone number is a string of alphanumerics and punctuation that uniquely identifies a
telephone service terminal, including extension number.</P></EXPLAIN>
<MODEL><STRING></STRING></MODEL>
</ELEMENTTYPE>

<H3>Fax</H3>
<ELEMENTTYPE NAME="fax">
<EXPLAIN><TITLE>Fax Number</TITLE>
<P>A fax number is a string of alphanumerics and punctuation that uniquely identifies a fax
service terminal.</P>
</EXPLAIN>
<MODEL><STRING></STRING></MODEL>
</ELEMENTTYPE>

<H3>Email</H3>
<ELEMENTTYPE NAME="email">
<EXPLAIN><TITLE>Email Address</TITLE>
<P>An email address is a datatype-constrained string that uniquely identifies a mailbox on a
server.</P></EXPLAIN>
<MODEL><STRING DATATYPE="email"></STRING></MODEL>
</ELEMENTTYPE>

<H3>Internet Address</H3>
<ELEMENTTYPE NAME="internet">
<EXPLAIN><TITLE>Internet Address</TITLE>
<P>An Internet address is a datatype-constrained string that uniquely identifies a resource on the
Internet by means of a URL.</P></EXPLAIN>
<MODEL><STRING DATATYPE="url"></STRING></MODEL>
</ELEMENTTYPE>

</DTD>
</SCHEMA>
```

Service Description Sample

```
<!DOCTYPE schema SYSTEM "bidl.dtd">
<SCHEMA>
<H1>Service Description Sample BID</H1>
<META
WHO.OWNS="Veo Systems"          WHO.COPYRIGHT="Veo Systems"
WHEN.COPYRIGHT="1998"           DESCRIPTION="Sample BID"
WHO.CREATED="*"                 WHEN.CREATED="*"
WHAT.VERSION="*"                WHO.MODIFIED="*"
WHEN.MODIFIED="*"               WHEN.EFFECTIVE="*"
WHEN.EXPIRES="*"                WHO.EFFECTIVE="*"
WHO.EXPIRES="*">
</META>

<PROLOG>
<XMLDECL STANDALONE="no"></XMLDECL>
<DOCTYPE NAME="service">
<SYSTEM>service.dtd</SYSTEM></DOCTYPE>
</PROLOG>

<DTD NAME="service.dtd">
<H2>Services</H2>

<H3>Includes</H3>
<!-- INCLUDE><SYSTEM>comments.bim</SYSTEM></INCLUDE -->

<H3>Service Set</H3>
<ELEMENTTYPE NAME="service.set">
<EXPLAIN><TITLE>Service Set</TITLE>
<SYNOPSIS>A set of services.</SYNOPSIS></EXPLAIN>
<MODEL>
<ELEMENT NAME="service" OCCURS="+"></ELEMENT>
</MODEL></ELEMENTTYPE>

<H3>Services Prototype</H3>
<PROTOTYPE NAME="prototype.service">
```

```
<EXPLAIN><TITLE>Service</TITLE></EXPLAIN>
<MODEL><SEQUENCE>
<ELEMENT NAME="service.name"></ELEMENT>
<ELEMENT NAME="service.terms" OCCURS="+"></ELEMENT>
<ELEMENT NAME="service.location" OCCURS="+"></ELEMENT>
<ELEMENT NAME="service.operation" OCCURS="+"></ELEMENT>
</SEQUENCE></MODEL>
<!-- ATTGROUP><IMPLEMENTS
HREF="common.attrib"></IMPLEMENTS></ATTGROUP -->
</PROTOTYPE>

<H3>Service</H3>
<INTRO><P>A service is an addressable network resource that provides interfaces to specific
operations by way of input and output documents.</P></INTRO>
<ELEMENTTYPE NAME="service">
<EXPLAIN><TITLE>Service</TITLE>
<P>A service is defined in terms of its name, the location(s) at which the service is available,
and the operation(s) that the service performs.</P></EXPLAIN>
<MODEL><SEQUENCE>
<ELEMENT NAME="service.name"></ELEMENT>
<ELEMENT NAME="service.location"></ELEMENT>
<ELEMENT NAME="service.operation" OCCURS="+"></ELEMENT>
<ELEMENT NAME="service.terms"></ELEMENT>
</SEQUENCE></MODEL>
</ELEMENTTYPE>

<H3>Service Name</H3>
<ELEMENTTYPE NAME="service.name">
<EXPLAIN><TITLE>Service Name</TITLE>
<P>The service name is a human-readable string that ascribes a moniker for a service. It may be
employed is user interfaces and documentation, or for other purposes.</P></EXPLAIN>
<MODEL><STRING></STRING></MODEL>
</ELEMENTTYPE>

<H3>Service Location</H3>
```

```
<ELEMENTTYPE NAME="service.location">
<EXPLAIN><TITLE>Service Location</TITLE>
<SYNOPSIS>A URI of a service.</SYNOPSIS>
<P>A service location is a datatype-constrained string that locates a service on the Internet by
means of a URI.</P></EXPLAIN>
<MODEL><STRING DATATYPE="url"></STRING></MODEL>
</ELEMENTTYPE>

<H3>Service Operations</H3>
<INTRO><P>A service operation consists of a name, location and its interface, as identified by
the type of input document that the service operation accepts and by the type of document that it
will return as a result.</P></INTRO>
<ELEMENTTYPE NAME="service.operation">
<EXPLAIN><TITLE>Service Operations</TITLE>
<P>A service operation must have a name, a location, and at least one document type as an
input, with one or more possible document types returned as a result of the operation.</P>
</EXPLAIN>
<MODEL><SEQUENCE>
<ELEMENT NAME="service.operation.name"></ELEMENT>
<ELEMENT NAME="service.operation.location"></ELEMENT>
<ELEMENT NAME="service.operation.input"></ELEMENT>
<ELEMENT NAME="service.operation.output"></ELEMENT>
</SEQUENCE></MODEL>
</ELEMENTTYPE>

<H3>Service Operation Name</H3>
<ELEMENTTYPE NAME="service.operation.name">
<EXPLAIN><TITLE>Service Operation Name</TITLE>
<P>The service operation name is a human-readable string that ascribes a moniker to a service
operation. It may be employed in user interfaces and documentation, or for other
purposes.</P></EXPLAIN>
<MODEL><STRING></STRING></MODEL>
</ELEMENTTYPE>

<H3>Service Operation Location</H3>
```

<INTRO><P>The service location is a network resource. That is to say, a URI.</P></INTRO>
<ELEMENTTYPE NAME="service.operation.location">
<EXPLAIN><TITLE>Service Operation Location</TITLE>
<SYNOPSIS>A URI of a service operation.</SYNOPSIS>
<P>A service operation location is a datatype-constrained string that locates a service operation on the Internet by means of a URL.</P></EXPLAIN>
<MODEL><STRING DATATYPE="url"></STRING></MODEL>
</ELEMENTTYPE>

<H3>Service Operation Input Document</H3>
<INTRO><P>The input to a service operation is defined by its input document type. That is, the service operation is invoked when the service operation location receives an input document whose type corresponds to the document type specified by this element.</P>
<P>Rather than define the expected input and output document types in the market participant document, this example provides pointers to externally-defined BIDs. This allows reuse of the same BID as the input and/or output document type for multiple operations. In addition, it encourages parallel design and implementation.</P></INTRO>
<ELEMENTTYPE NAME="service.operation.input">
<EXPLAIN><TITLE>Service Operation Input</TITLE>
<SYNOPSIS>Identifies the type of the service operation input document.</SYNOPSIS>
<P>Service location input is a datatype-constrained string that identifies a BID on the Internet by means of a URI.</P>
</EXPLAIN>
<MODEL><STRING DATATYPE="url"></STRING></MODEL>
</ELEMENTTYPE>

<H3>Service Operation Output Document Type</H3>
<INTRO><P>The output of a service operation is defined by its output document type(s). That is, the service operation is expected to emit a document whose type corresponds to the document type specified by this element.</P></INTRO>
<ELEMENTTYPE NAME="service.operation.output">
<EXPLAIN><TITLE>Service Operation Output</TITLE>
<SYNOPSIS>Identifies the type of the service operation output document.</SYNOPSIS>
<P>Service location output is a datatype-constrained string that identifies a BID on the Internet by means of a URI.</P>

```
</EXPLAIN>
<MODEL><STRING DATATYPE="url"></STRING></MODEL>
</ELEMENTTYPE>

<H3>Service Terms</H3>
<INTRO><P>This is a simple collection of string elements, describing the terms of an
agreement.</P></INTRO>
<ELEMENTTYPE NAME="service.terms">
<EXPLAIN><TITLE>Service Terms</TITLE>
<SYNOPSIS>Describes the terms of a given agreement.</SYNOPSIS>
</EXPLAIN>
<MODEL><STRING DATATYPE="string"></STRING></MODEL>
</ELEMENTTYPE>

</DTD>
</SCHEMA>
```

A service DTD schema is extended as follows and sold at the service type element of a common business language repository.

```
<!ELEMENT service.type EMPTY>
<!ATTLIST service.type
        service.type.name (
        catalog.operator
        | commercial.directory.operator
        | eft.services.provider
        | escrower
        | fulfillment.service
        | insurer
        | manufacturer
        | market.operator
        | order.originator
        | ordering.service
        | personal.services.provider
```

```
            | retailer
            | retail.aggregator
            | schema.resolution.service
            | service.provider
            | shipment.acceptor
            | shipper
            | van
            | wholesale.aggregator
            ) #REQUIRED
            %common.attrib;
  >
```

The above-mentioned service type element illustrates the content form which enables any one discernment of the effective service type in the interpretation information held by business interface definition and this example. Another interpretation information includes data typing like the element <H3> </H3> Internet Address expressed with a data type "string", for example including content form "url." Another interpretation information includes mapping of the code to an element of a list like the element <H3> </H3> State which includes code mapping to the country in a file "COUNTRY.US.SUBENTITY", for example.
The service description for which market authorized-personnel DTD refers specifies the document which service accepts and generates about the competition of service. A basic service description is indicated below as XML document transact.dtd.
transact.dtd models a transaction description like an invoice, or the description of exchange of a value. This document type supports many uses, therefore a transaction description element has an attribute which enables a user to distinguish an invoice, achievements, the proposal of sale, the request of the market price, etc. Although exchange is performed between two or more authorized personnel and got, only proposal people and two partners are made to appear. The everybody are expressed by the pointer to the document corresponding to market authorized-personnel DTD outlined above.Accepting the proposal of sale has a partner's arbitrary pointer. An exchange description is described by next tranprim.mod and includes a price and a subtotal. It must have a fee applied to a transaction as a whole following an exchange description, and the whole fee must be given. Therefore, transaction description schema document transact.dtd of this example is expressed below.

```
<!-- transact.dtd Version: 1.0 -->
<!-- Copyright 1998 Veo Systems, Inc. -->


...


<!ELEMENT transaction.description (meta?, issuer.pointer,
        counterparty.pointer?, exhange.description+, general.charges?,
        net.total?)>
<!ATTLIST transaction.description
        transaction.type (invoice | pro.forma | offer.to.sell | order
                | request.for.quote | request.for.bid
                | request.for.proposal | response.to.request.for.quote
                | response.to.request.for.bid
                | response.to.request.for.proposal) "invoice"
        %common.attrib;
        %altrep.attrib;
        %ttl.attrib;
>
```

Representation market authorized personnel and service DTD are generated according to the above-mentioned definition, and are as follows.
Market authorized-personnel DTD

```
<!ELEMENT business (party.name+, address.set) >
<!ATTLIST business business.number CDATA #REQUIRED
```

```
                >
    <!ELEMENT party.name (#PCDATA )>
    <!ELEMENT city (#PCDATA )>
    <!ELEMENT internet (#PCDATA )>
    <!ELEMENT country (#PCDATA )>
    <!ELEMENT state (#PCDATA )>
    <!ELEMENT email (#PCDATA )>
    <!ELEMENT address.physical  (street , city , state , postcode? , country) >
    <!ELEMENT telephone (#PCDATA )>
    <!ELEMENT person  (party.name+ , address.set) >
    <!ATTLIST person SSN   CDATA   #IMPLIED
                >
    <!ELEMENT fax (#PCDATA )>
    <!ELEMENT street (#PCDATA )>
    <!ELEMENT address.set  (address.physical , telephone* , fax* . email* . internet*) >
    <!ELEMENT postcode (#PCDATA )>
    <!ELEMENT market.participant  (business | person) >
```

Service DTD

```
  <!ELEMENT service.location (#PCDATA )>
  <!ELEMENT service.terms (#PCDATA )>
  <!ELEMENT service.operation.name (#PCDATA )>
  <!ELEMENT service.operation  (service.operation.name , service.operation.location ,
  service.operation.input , service.operation.output) >
  <!ELEMENT service  (service.name , service.location , service.operation+ , service.terms) >
  <!ELEMENT service.operation.input (#PCDATA )>
  <!ELEMENT service.operation.location (#PCDATA )>
  <!ELEMENT service.name (#PCDATA )>
  <!ELEMENT service.set (service+ )>
  <!ELEMENT service.operation.output (#PCDATA )>
```

An example of the document generated by transact.dtd continues.

```
<?xml version="1.0"?>
<!-- rorder.xml Version: 1.0 -->
<!-- Copyright 1998 Veo Systems, Inc. -->

<!DOCTYPE transaction.description SYSTEM "urn:x-
veosystems:dtd:cbl:transact:1.0:>
<transaction.description transaction.type="order">
<meta>
 <urn?urn:x-veosystems:doc:00023
 </urn>
        <thread.id party.assigned.by="reqorg">FRT876
        </thread.id>
</meta>
        <issuer.pointer>
                <xll.locator urllink="reqorg.xml">Customer
                Pointer
                </xll.locator>
        </issuer.pointer>
        <counterparty.pointer>
                <xll.locator urllink="compu.xml">Catalog entry owner
                pointer
                </xll.locator>
        </counterparty.pointer>
<exchange.description>
<line.item>
        <product.instance>
        <product.description.pointer>
                <xll.locator urllink="cthink.xml">Catalogue Entry    Pointer
                </xll.locator>
        </product.description.pointer>
        <product.specifics>
        <info.description.set>
<info.description>
 <xml.descriptor>
        <doctype>
```

```
          <dtd system.id="urn:x-veosystems:dtd:cbl:gprod:1.0"/>
          </doctype>
          <xml.descriptor.details>
          <xll.xptr.frag>DESCENDANT(ALL,os)STRING("Windows
95")
          </xll.xptr.frag>
          <xll.xptr.frag>DECENDANT(ALL,p.speed)STRING("200")
          </xll.xptr.frag>
          <xll.xptr.frag>DESCENDANT(ALL,hard.disk.capacity)
                    STRING("4")
          </xll.xptr.frag>
          <xll.xptr.frag>DESCENDANT(ALL,d.size)STRING("14.1")
          </xll.xptr.frag>
          </xml.descriptor.details>
 </xml.descriptor>
</info.description>
          </info.description.set>
          </product.specifics>
<quantity>1
</quantity>
          </product.instance>
<shipment.coordinates.set>
<shipment.coordinates>
<shipment.destination>
                    <address.set>
          <address.named>SW-1
          </address.named>
          <address.physical>
                    <building.sublocation>208C</building.sublocation>
                    <location.in.street>123
                    </location.in.street>
                    <street>Frontage Rd.
                    </street>
                    <city>Beltway
                    </city>
```

```
                        <country.subentity.us
                        country.subentity.us.name="MD"/>
                        <postcode>20000
                        </postcode>
                </address.physical>
                <telephone>
                        <telephone.number>617-666-2000
                        </telephone.number>
                        <telephone.extension>1201
                        </telephone.extension>
                </telephone>
                </address.set>
        </shipment.destination>

        <shipment.special>No deliveries after 4 PM</shipment.special>
        </shipment.coordinates>
        </shipment.coordinates.set>
                <payment.set>
                <credit.card
                issuer.name="VISA"
                instrument.number="3787-812345-67893"
                expiry.date="12/97"
                currency.code="USD"/>
                <amount.group>
                        <amount.monetary currency.code="USD">3975
                        </amount.monetary>
                </amount.group>
                </payment.set>
        </line.item>
        </exhange.description>
        </transaction.description>
```

[0046]
Therefore, a market participant identifies itself and this invention provides the art of enabling it to identify the type of the output document which is trying to conduct the type and dealings of input documents. The specific method by which the contents supported by this kind of document are processed by other parties or local parties in these dealings is not related to

business-related establishment, and related to trading.
[0047]
Drawing 3 is a schematic diagram of the participant node in the network by this invention. The node illustrated by drawing 3 contains the network interface 300 combined with a communication network in the port 301. This network interface is combined with the document purser 301. The purser 301 supplies the logical structure from an input document to the translator module 302, and the translator module 302, It changes into the form which can use an input document by a host trading system, and is a thing for [the] changing the output of a host processor into the output document form in a business interface definition conversely for transmission to the destination. The purser 301 and the translator 302 answer the business interface definition memorized by the participant module 303.
[0048]
The output data structure from the translator 302 is supplied to the dealing process front end 304 with the event in which signaling is done by the purser 301. The front end 304 in one example consists of a similar interface of a JAVA virtual machine suitable for communicating among various nodes in a network, or others. The dealing process front end 304 answers the event directed by the purser 301 and the translator 302, and carries out routing of the input data to the suitable function in the enterprise system and network with which the participant was combined. In this way, the dealing process front end 304 in the example of drawing 3, It is combined with the commercial function 305, the database function 306, accounting, and the enterprise function of others like the billing 307, It is combined with the specific event listeners and the processor 308 which were designed answer the event directed by the purser.
[0049]
The purser 301 receives the document of a purchase order [as / in the above-mentioned example specified according to a business interface definition], or others, The set of the event recognized by the local processing-of-transactions architecture like the set of the JAVA event for a JAVA virtual machine is generated.
[0050]
The purser of this invention is separated from the program which carries out the listening of the event based on a receiving document. The perfect document which fills various markup piece or the specific specification in a receiving document works as a command for the listening function which makes processing start. In this way, a listening program carries out business logic relevant to documents information. For example, the program relevant to an address element may be a code which corroborates a postal code by checking a database. These listeners join an event by registering with a document router. A document router dispatches a suitable event to all the members who are interested in them.
[0051]

For example, the purchase order specified as mentioned above is supervised by the program which carries out listening, and deals in the event generated by the purser who will connect a document or its contents to the order entry program. An inventory is checked using a certain program by receipt of the product description in the purchase order. By receipt of the address information in a purchase order, the availability of service for delivery is checked using a certain program. The buyer information field in a document can offer similar processing which is based on checking an order history about credit value, or getting to know the identity of a promotion or consumers using various processes.
[0052]
Complex listeners are generated as PURIMICHIBU listeners' configuration, and get. For example, including the list listeners whom a certain purchase order listeners mentioned above, using the list listeners, Japanese lacquer and list members are used about these selves, and it deals in them. The application that listeners run should be cautious of what is not a NECHIBU XML process or a NECHIBU JAVA process. An object is changed into the format needed by reception transformer application in these cases. When the application completes processing, the output is returned to an XML format for the communication to other nodes in a network.
[0053]
As market participant document type description and dealings document type description which were mentioned above, there is skimmer tic mapping for the logic element in a document, and there is a markup language based on natural language. A natural language markup and the natural language attribute of others of XML make easy use of the XML type markup language for the specification of a business interface definition, service descriptions, an input, and description of an output document.
[0054]
A business interface definition to memorize in addition, the participant module 303, The compiler used for editing the data structure of the object used by the dealing process front end 304 corresponding to the logical structure in an input document or others, or editing the translator 302 is included. In this way, when a participant changes or it is updated along with change of the dealings related to a business interface definition in a participant, the translator 302 and the purser 301 are kept automatic to update.
[0055]
In a desirable system, the set of a JAVA event, As the glove model of SGML, and the Lord, it is generated by the compiler corresponding to the element structure information set of the standard extended by a property set about each element (International Standard ISO/IEC 10179:1996 (E).) Information Technology--Processing Language--Document Style Semantic and Specification Language (DSSSL). Returning an XML document to the set of an event for a process differs from the usual model of syntax analysis that a purser output is maintained as

an internal data structure. By changing the element of an XML document into a JAVA event or other programming structure suitable for using it by the processing-of-transactions front end of each node, The rich functionality in the node using the document traded is made possible.
[0056]
In this way, the dealing process front end 304 can operate in Publish and the subscribe architecture which enable it to add a listeners program [nothing / or / the knowledge of other listening programs in a system] new without impact. Each listeners 305, 306, 307, and 308 in drawing 3 maintain the cue with which the front end 304 commands an event. Thereby, many listeners can deal with an event now at the pace of these selves.
[0057]
According to this invention, the application that listeners run does not need to be the NECHIBU XML function or NECHIBU function adjusted to the format of an input document further again. rather, when the dealing process front end 304 is a JAVA interface, these listeners are JAVA functions and may be functions which obtain and carry out or run according to the peculiar processing-of-transactions architecture. In these cases, an object is changed into the format needed by receiving application, and it deals in it. When listeners' application finishes, the output is returned to the format of a document which is specified by the business interface definition in the module 303. In this way, the translator 302 is directly combined with the network interface 300 in order to supply the combined document as an output.
[0058]
The listeners combined with the processing-of-transactions front end may contain the listeners for the attribute memorized by the listeners for an input document, the listeners for the specific element of an input document, and the specific element of the input document. Operation with flexibility with various dealing processes in the specific node for filtering an input document and answering it by this, is attained.
[0059]
Drawing 4 has illustrated the process for receiving and processing the input document for the system of drawing 3. In this way, a process starts by receiving a document by a network interface (Step 400). A purser answers a business interface definition and identifies a document type (401). The business interface definition which has memorized DTD for a document in the XML format is used, and the syntax of a document is analyzed (Step 402). Next, the element and attribute of a document are changed into a host's format (Step 403). XML logic structure is changed into the JAVA object which supports the data of an XML element with the method relevant to the data like get and a set function in this example. Next, a host object is transmitted to a host processing-of-transactions front end (Step 404). These objects answer the event directed by a purser and the translator, and routing is carried out to a process. The process of receiving the element of a document is performed and generates an

output (Step 405). The output is changed into the format of an output document which is defined by business interface definition (Step 406). In this example, the translation progresses to the form of an XML document from the form of a JAVA object. Finally, an output document is transmitted to the destination through a network interface (Step 407).
[0060]
Drawing 5 is a detailed figure of the event generation machine / event listeners mechanism for the system of drawing 3. Generally, the technique shown in drawing 5 is improvement of a JAVA JDK1.1 event model. Three kinds of objects are taken into consideration in this model.The object of the 1st kind is an event object including the information about generating of an event. There is the number of the kinds of event object corresponding to the complete range from which the event which can be generated differs. The object of the 2nd kind is an event generation machine which supervises activity and generates an event object, when something happens. The 3rd is event listeners who hear the event object generated by the event generation machine. Generally event listeners hear specific event generation machines, such as a mouse click on a specific window. Event listeners call the method of "adding event listeners" on an event generation machine. This model can be fitted to the environment of drawing 3, it answers walking along it and analyzing the syntax of the graph of an object which is represented by the XML document, and an object is generated.
[0061]
The system shown in drawing 5 has general XML parser 500. Such a purser can be realized using a standard recall model. When an syntax-analysis event occurs, a purser calls a specific method within an application object, and hands the suitable information in a parameter. Therefore, the single application 501 exists with a purser. Application sends it to a number of event listeners identified by self so that the information which a purser provides may be packed in an XML event object and it may be shown by the block 502. The group of the event 502 is independently from a purser thoroughly. the event 502 -- the listeners of what kind of number on the machine of what kind of number, and the thread of what kind of number -- being also alike -- it can be supplied. The event is based on the element structure information set (ESIS) in one alternative form. Therefore, these consist of a list of important viewpoints like recognition of document structures, such as a start of an element, and an end, or an attribute. Generally an XML (and SGML) purser uses ESIS structure as a default set of information for a purser to return to the application.
[0062]
The specialized ESIS listeners 503 are combined with the group of the event 502. These listeners 503 realize ESIS listeners API, and hear 1 or all the XML events from the generator from plurality. The one element event generation machine 504 is the specialized ESIS listeners, and is also an XML event generation machine. The listeners are objects which are

interested only in the event to an element specific type. For example, in the HTML environment, listeners have interest only into the portion of the document between the directed list, i.e., <OL>, and a </OL> tag. In other examples, listeners hear only a "party.name" element or a "service.name" element from the document of the upper example according to a common business language, In order to ensure that an element carries the data which is in agreement with diagrammatic mapping for an element, an element is processed, and it reacts according to the process needed in a receipt node.

[0063]

This enables a system to have a small object which hears only the particular part of the document [like] which totals a price. Since listeners add these selves from a generator or it can remove, the listeners who hear only for example, the <HEAD> portion of a HTML document can also exist. For this reason and the reason of the high repetitiousness of an XML document, writing a high target code and concurrence listeners can be written. For example, the <OL> listeners can dissociate thoroughly and can set the <LI> listeners to the way the <UL> (list which is not directed) listeners set up the <LI> listeners. Another listeners who look for the database which uses the listeners and the same input which generate a figure user interface as an alternate method can be created. Therefore, contrary to the completed data structure which application inspects one [at a time] at once, a document is dealt with as a program executed by listeners. When application is written by this method, in order to perform application, there is no necessity of having all the documents in a memory.

[0064]

The next listeners combined with the group of the event 502 are the attribute filters 505. The attribute filter 505 is an attribute event generation machine according to an ESIS listeners model like the element filter 504. The listeners for an attribute filter specify the attribute in which it gets interested, and receive the event for what kind of element with the specified attribute. For example, a font manager will receive the event only for an element with a font attribute like <P FONT="Times Roman"/P>.

[0065]

The element event generation machine 504 supplies such an element object, in order to specialize the element listeners 504A.

[0066]

The attribute event generation machine 505 supplies an attribute event object to the attribute listeners 505A. Similarly, attribute objects are supplied to "architecture" in the meaning which carries out SGML/XML modification from the document form of 1 to an exception using an attribute. Therefore, the architecture of 505B enables it to identify a specific attribute with a specific name. Only an element with the defined attribute becomes a portion of an output document, and the name of the element in an output document is a value of the attribute in an

input document. For example, the architecture 505B is HTML and a string, if <PURCHASES HTML="OL"> <ITEM HTML="LI"> <NAME HTML= "B"> STUFF</NAME> <=[PRICE HTML] "B" >123</PRICE> </ITEM> </PURCHASE> translation of is done, <OL> <LI>STUFF </B><B> </B>123 It is set to </LI </OL>> and is right HTML.
[0067]
The following module combined with the group of the event 502 is the tree (Thurs.) builder 506. A tree builder takes the stream of an XML event and tree (Thurs.) expression of the document which is downward is generated. One desirable form of the tree builder 506, According to the specification of W3C (http://www.w3.org/TR/1998/WD-DOM-19980720/introduction.html), the document object model DOM object 507 is generated. Although the listeners in an event stream can use it for processing a great portion of demand, It is useful for tree form to support the data structure in the memory which can carry out multiple-times generating of the event stream same from there where the syntax of creation of a new document, for example, a document, is analyzed for a node many times for a re-********** reason in order to support the question of the circumference of a document. The specialized builder 508 is combinable with the tree builder 506, in order to make a subtree with a special portion of a document which suits specific operation.
[0068]
In addition to answering the document to input, other sources of the XML event 502 can be supplied. Therefore, the event stream 510 is generated walking on the tree of a DOM object, and by acting as a recurrence student of the event stream of the origin created when the syntax of a document was analyzed and it was. This enables a system to give the appearance by which the syntax of a document is analyzed several times.
[0069]
The idea of an object of walking along a tree and generating the stream of an event is generalizable to what kind of tree of the object which can be asked over the tree of a DOM object. Therefore, the application which walks along the tree of the JAVA flowbean component 513 may be satisfactory for the JAVA walker 512. A walker walks on all the fields currently exhibited and methods. A walker memorizes the locus of the object which it already visited, in order to secure not entering into a cycle without the end. The JAVA event 514 is a type of the event generated by the JAVA walker 512. This includes now most kinds of information which can be pulled out from an object. This is a JAVA equivalent of ESIS.
And ** is made possible although it is at applying the same program technique as having been applied to XML to a general JAVA object, especially JAVA BINZU.

[0070]
The XML event generation machine 515 comprises JAVA listeners and a JAVA event

generation machine from JAVA. This receives the stream 514 of an event from the JAVA walker 512, and translates what was chosen in order to supply a JAVA object as an XML document. In a desirable example, the event generation machine 515 uses JAVA BINZU API. Each seen object serves as an element and the element name is the same as a class name. Within the element, each ****** rare ****** also serves as an element, and the contents are the values returned by stirring up a method. These walk by turns that it is the arrangement of an object or an object.

[0071]

Drawing 6 is the schematic diagram of specific application built on the framework of drawing 5. This application incorporates the XML document 600 and applies this to a purser / generator (syntax-analysis routine generation system) 601. The ESIS event 602 is generated and the attribute generator 603 and the tree builder 604 are supplied. An attribute generator is equivalent to the generator of drawing 5. This generator 505 is sent to "architecture" 505B for translating an event into a HTML output from an XML input, for example. These events are parallel, are processed so that it may be shown by the block 605, and they are processed by listeners. Listeners' output is supplied to the document writer 606, and is returned and translated into an XML format for an output. Therefore, for example, this application shown in drawing 6 takes in an XML document, and outputs the HTML document which has a certain form. This form is sent to a browser next, and that result is returned and changed into XML. For this employment, an architecture concept provides mapping to HTML from XML. A thing for the three architecture contained in drawing 6 to give the structure of a table and a HTML document like a list, The 2nd thing that specifies a document like the label for the input fields on a browser document which should be displayed, and the 3rd thing that describes the input field itself are included. The element of an XML document which is needed for maintaining XML document structure serves as the invisible field within HTML form. This is useful, although it is used when an XML document is reconstructed from the information which a client puts into the HTTP mailbox message returned to a server. The listeners who hear these events output the event for HTML documents. This document goes to a document writer object. A document writer object hears an XML event and returns them to an XML document. Document writer objects are the listeners to all the element generators which are hearing the architecture in this example.

[0072]

The composition of the treatment module illustrated by drawing 5 and 6 expresses the purser for the systems of drawing 3, and a certain embodiment of the transaction process front end. An extremely supple interface is given, by this, the XML document to input or other structure document formats can be answered, and a diverse (change) transaction process can be performed so that he can understand.

[0073]
Drawing 7 is illustrating the node similar to drawing 3 except for the business interface definition builder module 700 being included. Therefore, the system of drawing 7 contains the network interface 701, the document purser 702, and the document translator 703. The translator 703 outputs the output to the transaction processing front end 704. This front end is combined with a listening function like the function 705 on commerce, the database 706, the company function 707, other general listeners, and the processor 708. The business interface builder 700 contains the process for reading user yne TAESU, a common business library CBL repository, and complementary business interface definition **, and a compiler so that it may be illustrated by drawing 7. A user interface is used for assisting the attempt in construction of a business interface definition depending on the capability to read the definition of a common business library repository and a complementary business interface. Therefore, the input document of a definition of a complementary business interface can be specified as an output document of a specific transaction, and the output document of a definition of a complementary business interface can specify it as an input like a transaction process. It is ****** that the definition of a transaction business interface consists of same methods using the ingredient chosen from the CBL repository. Use of a CBL repository promotes use of the logical structure in construction of a definition of a standardized document format like an above-mentioned illustration proposal (bid1) document, and a business interface immediately employable by other human beings in a network, and interpretation information.
[0074]
The business interface definition builder 700 generates the object which should be processed by the translator according to the translator 703 and the host transaction processing architecture, and also contains the compiler used in order to manage the parsing function 702.
[0075]
Drawing 8 is a heuristic graph showing the logical structure memorized in the repository in the business interface definition builder 700. Therefore, the repository memory showing the party business interface definition 800 contains consumer BID801, catalog house BID802, warehouse BID803, and action house BID804, for example. Therefore, the new participant in an on-line market chooses one of the business and the standardized BID(s) which suit best as a basic interface definition. A repository memorizes the service business interface definition 805 of a lot. For example, order entry BTD806, order tracking BID807, order fulfillment BID808, and catalog service BID809 are memorizable. When the new participant in a market builds the definition of a business interface, the business interface definition of the standardized service which is memorized in a repository can be chosen.
[0076]
In addition to a party and the service BID, an input and the output document BID are

memorized by the repository directed by the field 810. Therefore, purchase order BID811, bill BID812, request-for-quotation BID813, product available report BID814, and order letter voice BID815 are memorizable to a repository.
[0077]
In addition to the definition of the business interface specified as a definition of a document gestalt according to XML, in a suitable system, a repository memorizes interpretation information with the gestalt of a semantic map which is directed by the field 816. Therefore, the semantic map used since the weight 817, the currency 818, the size 819, the product identification 820, and the product feature 821 are specified is memorizable in a repository. Interpretation information prepares the data structure within the logical structure of a document for carrying out classification.
[0078]
The logical structure used when the definition of a business interface is constituted is memorized by the repository pointed out by the block 822. Therefore, the form for giving the address information 823, the form for giving the price information 824, and the form for giving the period 825 of a relation contractually can be provided. When a network is extended, a CBL repository is extended similarly and standardization makes easier the definition of that a new participant joins and a business interface.
[0079]
Drawing 9 is illustrating the process of building the business interface definition which uses the system of drawing 7. A process is started by displaying a BID builder graphical interface on a user (Step 900). A system accepts the user input which identifies the participant, service, and documents information which were generated by the graphical interface (Step 901).
[0080]
Next, what kind of the reference logical structure, interpretation information, a document definition, and/or a service definition answer a user input, and are searched from a repository via a graphical user interface (Step 902). The ingredient of a complementary business interface definition or the definition of a business interface at the following step by the search engine by which custom-made ** was carried out, the web browser, or other methods. It is accessed by other participants in the network selected via the user input (Step 903). The definition of the document to a participant is generated using the collected information (Step 904). The translator to each mapper from a document to the host and document from a host is generated by the compiler (Step 905). The host architecture data structure corresponding to a definition is generated by the compiler (Step 906). and by post sending to a website or other places, the definition of the generated business interface is post sent on a network, and becomes accessible at other nodes in a network -- it is carried out. (Step 907).
[0081]

The definition of a business interface tells the document used for telling the on-line service which the company offered to the potential commercial partner, and carrying out these services. Therefore, service is defined by the definition of a business interface by the document which service receives and generates. This is illustrated by the following fragmentation of the definition of XML service.
[0082]

```
<service>
<service.name>Order Service</service.name>
<service.location>ww.veosystems.com/order</service.location>
<service.op>
<service.op.name>Submit Order</service.op.name>
<service.op.inputdoc>www.commerce.net/po.dtd</service.op.inputdoc>
<service.op.outputdoc>
         www.veosystems.com/invoice.dtd</service.op.outputdoc>
</service.op>
<service.op>
<service.op.name>Track Order</service.op.name>
<service.op.inputdoc>www.commerce.net
         /request.track.dtd<service.op.inputdoc>
<service.op.outputdoc>
         www.veosystems.com/response.track.dtd<service.op.outputdoc>
</service.op>
</service>
```

[0083]
The service as for which this XML fragmentation (fragment) consists of two processings, i.e., the processing from which one acquires an order (order), and another define the processing which follows it. Each definition will express the contract (contract) or promise (promise) for performing service, if a specific Web address is shown a just demand. . Are placed into repository (repository) as which what is local as for order service, or was memorized in the

registry of the whole industry on a network here may be sufficient. The document (document) which suits standard "po.dtd" DTD (Document Type Definition) and which is inputted is needed. Supposing one node can perform the order, the node will return the document with a local definition which suits customized "invoice.dtd." Actually, the company is making a promise that anyone who can submit the purchase order which suits the XML specification which it declares performs business. Prior adjustment is unnecessary.

[0084]

DTD is, the formal specification, i.e., the grammar, for the document of a predetermined mold. It describes elements, those attributes, and the state where they must be displayed.

For example, a purchase order contains typically the name of a buyer and a vender and an address, the product description of a lot, a price, and the related terms of sales like a delivery day. In electronic data interchange (Electronic Data Interchange, EDI), it is X12, for example. 850 specification is a model used ordinarily because of a purchase order.

[0085]

Developing an XML document model is promoted from the common ingredient (reusable semanticcomponents) of a reusable meaning in many business fields by the repository. Even if such a document completely differs in those appearance, he can understand it from those common message elements. This is a role of a common business library repository (Common Business Library repository).

[0086]

A common business library repository consists of an information model for the general business concept containing the following.

- A business description primitive like a company, service, and a product (business description primitive), - the measured value (standard measurements) a catalog, a purchase order and a business gestalt like an invoice, and standard for -, time, a position, and a classification code -- come out.

[0087]

This information is expressed as an extensible and exhibited XML configuration block (XML building block) which can be customized and assembled, in order that a company may develop XML application promptly. A very small (atomic) CBL element A country, currency, an address (address), And the messaging standard and established practice (industry messaging standards and conventions) of the industrial world like the standard ISO code for time are performed. CBL semantics (semantics) of a low is obtained also from analysis of the metadata composition (metadata framework) to which it was shown for an Internet resource like the Dublin core (Dublin Core).

[0088]

These configuration blocks are used for the element of the following level, and like OTP (Open

Trading Protocol) and OBI (Open Buying on the Internet), Although used not only by the business gestalt used by the Internet standard which appeared but by X12 EDI processing, a fundamental business gestalt [like] is also performed.
[0089]
The chief aim of CBL is in a function and information common to all the business fields (the business description primitive; catalog and purchase order like a company, service, and a product, and a business gestalt like an invoice; standard measured value, time, a position, a classification code). CBL is dependent on the standard for the semantics in the case of being possible, or the established practice of the industrial world (for example, the rule which specifies the "month/day/year" in the United States to the "day/month/year" in Europe is coded in a separate CBL module).
[0090]
CBL is a language used in order to design application. It is designed mediate between the "document world" of XML, the "programming world" of JAVA, or the gap between other treatment process architecture. Schema (schema) materializes the thought of "programming by a document" whose detailed formal specification of a document type is the master sauce (master source) in which the gestalt to which versatility relates is generated from there. These gestalten contain XML DTD for the conversion program between CBL, a JAVA object, and the instance of XML and a corresponding JAVA object, and a support document.
[0091]
CBL makes the single sauce that almost all the portions of a system are automatically generated by the compiler from there. CBL functions by extending SGML/XML by which normal use is carried out, in order to define the structure of a specific document type officially so that the specification of the semantics relevant to the element of each information and an attribute may be included. The character type of the group in SGML/XML to which it was restricted (most) is extensible in order to declare arbitrary kinds of data type.
[0092]
This is the fragmentation from the CBL definition for the "datetime" module.

```
<!-- datetime.mod Version: 1.0-->

(!-- Copyright 1998 Veo Systems, Inc.--)

...

<! ELEMENT year (#PCDATA)>
<! ATTLIST year
        schema CDATA #FIXED ”urn:x-veosystems:stds:iso:8601:3.8”
>

<! ELEMENT month (#PCDATA)>
<! ATTLIST month
        schema CDATA #FIXED ”urn:x-veosystems:stds:iso:8601:3.12”
>

...
```

[0093]
ELEMENT"year" is defined as character data and a related "schema" attribute by this fragmentation, and defines the schema for "year" as character data becoming the section 3.8 of ISO8601 standard by it.
[0094]
This "datetime"CBL module is defined as an instance of schema DTD in practice. First, a module name is defined. Next, "YEAR" of a "datetime" element is combined with the semantics of ISO8601 standard.
[0095]

```
<! DOCTYPE SCHEMA SYSTEM ”schema.dtd”>
<SCHEMA><H1>Date and Time Module</H1>

...
<ELEMNTTYPE NAME=”year” DATATYPE=”YEAR”><MODEL>
     <STRING
DATATYPE=”YEAR”></STRING></MODEL>
<ATTDEF NAME=:schema:iso8601”DATATYPE=”CDATA”>
     <FIXED>3.8
```

```
Gregorian calendar</FIXED></ATTDEF></ELEMENTTYPE>

...
```

[0096]
The participant and the above-mentioned services module of a market of an example are also memorized in a CBL repository.
[0097]
The air waybill (Airbill) 1000 customizes general purchase order DTD1001, and is just going to define by adding the more concrete information about the shipment weight 1002 by drawing 10. From the CBL module for an address, time, currency, a vender, and a product description, the general purchase order 1001 reached far and wide, and was composed first. Thus, the implementation of XML commercial application is considerably promoted by using CBL. It becomes easy to carry out interconnection of the commercial application to a more important thing by CBL.
[0098]
XML is extended by the schema in CBL. By extension, emphasis typing (strong-typing) is added to an XML element so that the contents can be checked easily. For example, the element called <CPUclockspeed> can be defined as an integer which has value (valid value): with a just lot {100, 133, 166,200,233,266 Mhz}.The hierarchy of a class subclass also adds a schema so that instance generation of the information can be easily carried out from the definition of a class. For example, it can describe as a computer which has a tag of addition of laptop for the feature like the type of a display, and the life of a battery. By these and other extension, not only the automatic translation between XML, traditional object-orientation, and a relational data model but data input becomes easy.
[0099]
Thus, the actual instance of service which outlined completed BTD to a participant and the above, It performs through the compiler which makes DTD for JAVA BINZU (beans) corresponding to the logical configuration of a DTD instance, and the conversion code changed into JAVA and XML from XML from JAVA. In other systems, in order to make an object easy to use, a document is also generated for printing by the user for the display on a user interface.
[0100]
For example, JAVA BINZU generated by a market participant and above-mentioned service DTD, and a compiler is shown as follows (adding some corrections, since it is brief).

```
            import com.veo.vsp.doclet.meta.Document;
public class AddressPhysical extends Document {
            public static final String DOC_TYPE = "address.physical";
             String mStreet;
             String mCity;
            public final static int AK = 0;
            public final static int AL = 1;
            public final static int AR = 2;
            public final static int AZ = 3;
            public final static int CA = 4;
            ...

            public final static int WI = 48;
            public final static int WV = 49;
            public final static int WY = 50;
             int mState;
             String mPostcode;
            public final static int AD = 51;
            public final static int AE = 52;
            public final static int AF = 53;
            public final static int AG = 54;
            public final static int AI = 55;
            public final static int AM = 56;
            ...

             int mCountry;
            public AddressPhysical(){
                        super(DOC_TYPE);
            mStreet = new String();
            mCity = new String();
                        this.mState = -1;
            mPostcode = null;
                        this.mCountry = -1;
            }
            public AddressPhysical(String doc_type){
```

```
            super(doc_type);
        mStreet = new String();
        mCity = new String();
            this.mState = -1;
        mPostcode = null;
            this.mCountry = -1;
        }
        static public AddressPhysical initAddressPhysical(String iStreet,String iCity,int
iState,String iPostcode,int iCountry){
            AddressPhysical obj = new AddressPhysical();
            obj.initializeAll(iStreet, iCity, iState, iPostcode, iCountry);
            return obj;
        }

        public void initializeAll(String iStreet,String iCity,int iState,String iPostcode,int
iCountry){
            mStreet = iStreet;
            mCity = iCity;
            mState = iState;
            mPostcode = iPostcode;
            mCountry = iCountry;
        }
        public String getStreet(){
         return mStreet;
        }
        public String getStreetToXML(){
            if (getStreet() == null) return null;
            char [] c = getStreet().toCharArray();
            StringBuffer sb = new StringBuffer();
            for (int x = 0; x < c.length; x++){
             switch(c[x]){
             case '>':
              sb.append("&gt;");
              break;
             case '<':
                sb.append("&lt;");
                break;
              case '&':
                sb.append("&amp;");
                break;
              case '"':
                sb.append("&quot;");
                break;
              case '\'':
                sb.append("&quot;");
                break;
              default:
                if (Character.isDefined(c[x]))
                  sb.append(c[x]);
              }
            }
            return sb.toString();
    }
    public void setStreet(String inp){
            this.mStreet = inp;
    }
    public void streetFromXML(String n){
            setStreet(n);
    }
    public String getCity(){
      return mCity;
    }
    public String getCityToXML(){
            if (getCity() == null) return null;
            char [] c = getCity().toCharArray();
            StringBuffer sb = new StringBuffer();
            for (int x = 0; x < c.length; x++){
              switch(c[x]){
              case '>':
                sb.append("&gt;");
```

```
          break;
        case '<':
          sb.append("&lt;");
          break;
        case '&':
          sb.append("&amp;");
          break;
        case '"':
          sb.append("&quot;");
          break;
        case '\'':
          sb.append("&quot;");
          break;
        default:
          if (Character.isDefined(c[x]))
            sb.append(c[x]);
        }
      }
      return sb.toString();
}
public void setCity(String inp){
      this.mCity = inp;
}
public void cityFromXML(String n){
      setCity(n);
}
public int getState(){
  return mState;
}
public String getStateToXML(){
      switch (mState){
            case AK: return "AK";
            case AL: return "AL";
            case AR: return "AR";
            case AZ: return "AZ";
```

```
                ...

          }
   return null;
  }
  public void setState(int inp){
          this.mState = inp;
  }
  public void stateFromXML(String s){
          if (s.equals("AK")) mState = AK;
          else if (s.equals("AL"))mState = AL;
          else if (s.equals("AR"))mState = AR;
          else if (s.equals("AZ"))mState = AZ;
          ...

  }
  public String getPostcode(){
   return mPostcode;
  }
  public String getPostcodeToXML(){
          if (getPostcode() == null) return null;
          char [] c = getPostcode().toCharArray();
          StringBuffer sb = new StringBuffer();
          for (int x = 0; x < c.length; x++){
           switch(c[x]){
           case '>':
            sb.append("&gt;");
            break;
           case '<':
            sb.append("&lt;");
            break;
           case '&':
            sb.append("&amp;");
            break;
           case '"':
```

```
                        sb.append("&quot;");
                        break;
                      case '\'':
                        sb.append("&quot;");
                        break;
                      default:
                        if (Character.isDefined(c[x]))
                          sb.append(c[x]);
                    }
                  }
                  return sb.toString();
          }
          public void setPostcode(String inp){
                  this.mPostcode = inp;
          }
          public void postcodeFromXML(String n){
                  setPostcode(n);
          }
          public int getCountry(){
            return mCountry;
          }
          public String getCountryToXML(){
                  switch (mCountry){
                          case AD: return "AD";
                          case AE: return "AE";
                          case AF: return "AF";
                          ...

                  }
            return null;
          }
          public void setCountry(int inp){
                  this.mCountry = inp;
          }
          public void countryFromXML(String s){
```

```
                    if (s.equals("AD")) mCountry = AD;
                    else if (s.equals("AE"))mCountry = AE;
                    else if (s.equals("AF"))mCountry = AF;
                    else if (s.equals("AG"))mCountry = AG;
                    else if (s.equals("AI"))mCountry = AI;
                    ..

            }
}

package com.veo.xdk.dev.schema.test.blib;

import com.veo.vsp.doclet.meta.Document;
public class AddressSet extends Document {
            public static final String DOC_TYPE = "address.set";
            AddressPhysical mAddressPhysical;
            String [] mTelephone;
            String [] mFax;
            String [] mEmail;
            String [] mInternet;
            public AddressSet() {
                    super(DOC_TYPE);
                    this.mAddressPhysical = new AddressPhysical();
            mTelephone = null;
            mFax = null;
            mEmail = null;
            mInternet = null;
            }
            public AddressSet(String doc_type) {
                    super(doc_type);
                    this.mAddressPhysical = new AddressPhysical();
            mTelephone = null;
            mFax = null;
            mEmail = null;
            mInternet = null;
```

```
    }
    static public AddressSet initAddressSet(AddressPhysical iAddressPhysical.String []
iTelephone,String [] iFax,String [] iEmail,String [] iInternet){
            AddressSet obj = new AddressSet();
            obj.initializeAll(iAddressPhysical, iTelephone, iFax, iEmail, iInternet);
            return obj;
    }

    public void initializeAll(AddressPhysical iAddressPhysical,String [] iTelephone.String
[] iFax,String [] iEmail,String [] iInternet){
            mAddressPhysical = iAddressPhysical;
            mTelephone = iTelephone;
            mFax = iFax;
            mEmail = iEmail;
            mInternet = iInternet;
    }
    public AddressPhysical getAddressPhysical(){
     return mAddressPhysical;
    }
    public void setAddressPhysical(AddressPhysical inp){
            this.mAddressPhysical = inp;
    }
    public String [] getTelephone(){
     return mTelephone;
    }
    public String getTelephone(int index){
            if (this.mTelephone == null)
             return null;
            if (index >= this.mTelephone.length)
              return null;
            if (index < 0 && -index > this.mTelephone.length)
              return null;
            if (index >= 0) return this.mTelephone[index];
            return this.mTelephone[this.mTelephone.length + index];
    }
```

```
public String [] getTelephoneToXML(){
            String [] valArr = getTelephone();
            if (valArr == null) return null;
            String [] nvArr = new String[valArr.length];
            for (int z = 0; z < nvArr.length; z++){
            char [] c = valArr[z].toCharArray();
            StringBuffer st = new StringBuffer();
            StringBuffer sb = new StringBuffer();
            for (int x = 0; x < c.length; x++){
             switch(c[x]){
             case '>':
              sb.append("&gt;");
              break;
             case '<':
              sb.append("&lt;");
              break;
             case '&':
              sb.append("&amp;");
              break;
             case '"':
              sb.append("&quot;");
              break;
             case ''':
              sb.append("&quot;");
              break;
             default:
              if (Character.isDefined(c[x]))
               sb.append(c[x]);
             }
           }
           nvArr[z] = sb.toString();
           }
           return nvArr;
}
public void setTelephone(int index, String inp){
```

```
            if (this.mTelephone == null) {
                if (index < 0) {
                 this.mTelephone = new String[1];
                 this.mTelephone[0] = inp;
                } else {
                 this.mTelephone = new String[index + 1];
                 this.mTelephone[index] = inp;
                }
           } else if (index < 0) {
                        String [] newTelephone = new String[this.mTelephone.length + 1];
                        java.lang.System.arraycopy((Object)mTelephone, 0,
(Object)newTelephone, 0, this.mTelephone.length);
                        newTelephone[newTelephone.length - 1] = inp;
                        mTelephone = newTelephone;
                } else if (index >= this.mTelephone.length){
                        String [] newTelephone = new String[index + 1];
                        java.lang.System.arraycopy((Object)mTelephone, 0,
(Object)newTelephone, 0, this.mTelephone.length);
                        newTelephone[index] = inp;
                        mTelephone = newTelephone;
                } else {
                        this.mTelephone[index] = inp;
                }
        }
        public void setTelephone(String [] inp){
                this.mTelephone = inp;
        }
        public void telephoneFromXML(String n){
                setTelephone(-1, n);
        }
        public String [] getFax(){
         return mFax;
        }
        public String getFax(int index){
                if (this.mFax == null)
            return null;
          if (index >= this.mFax.length)
            return null;
          if (index < 0 && -index > this.mFax.length)
            return null;
          if (index >= 0) return this.mFax[index];
          return this.mFax[this.mFax.length + index];
}
public String [] getFaxToXML(){
          String [] valArr = getFax();
          if (valArr == null) return null;
          String [] nvArr = new String[valArr.length];
          for (int z = 0; z < nvArr.length; z++){
          char [] c = valArr[z].toCharArray();
          StringBuffer st = new StringBuffer();
          StringBuffer sb = new StringBuffer();
          for (int x = 0; x < c.length; x++){
           switch(c[x]){
           case '>':
             sb.append("&gt;");
             break;
           case '<':
             sb.append("&lt;");
             break;
           case '&':
             sb.append("&amp;");
             break;
           case '"':
             sb.append("&quot;");
             break;
          case ''':
             sb.append("&quot;");
             break;
          default:
            if (Character.isDefined(c[x]))
```

```
                    sb.append(c[x]);
                  }
                }
                nvArr[z] = sb.toString();
                }
                return nvArr;
        }
        public void setFax(int index, String inp){
            if (this.mFax == null) {
                if (index < 0) {
                 this.mFax = new String[1];
                 this.mFax[0] = inp;
                } else {
                 this.mFax = new String[index + 1];
                 this.mFax[index] = inp;
                }
            } else if (index < 0) {
                        String [] newFax = new String[this.mFax.length + 1];
                        java.lang.System.arraycopy((Object)mFax, 0, (Object)newFax, 0,
this.mFax.length);
                        newFax[newFax.length - 1] = inp;
                        mFax = newFax;
                } else if (index >= this.mFax.length){
                        String [] newFax = new String[index + 1];
                        java.lang.System.arraycopy((Object)mFax, 0, (Object)newFax, 0,
this.mFax.length);
                        newFax[index] = inp;
                        mFax = newFax;
                } else {
                        this.mFax[index] = inp;
                }
        }
        public void setFax(String [] inp){
                this.mFax = inp;
        }
```

```
public void faxFromXML(String n){
          setFax(-1, n);
}
public String [] getEmail(){
 return mEmail;
}
public String getEmail(int index){
          if (this.mEmail == null)
           return null;
          if (index >= this.mEmail.length)
            return null;
          if (index < 0 && -index > this.mEmail.length)
            return null;
          if (index >= 0) return this.mEmail[index];
          return this.mEmail[this.mEmail.length + index];
}
public String [] getEmailToXML(){
          String [] valArr = getEmail();
          if (valArr == null) return null;
          String [] nvArr = new String[valArr.length];
          for (int z = 0; z < nvArr.length; z++){
          char [] c = valArr[z].toCharArray();
          StringBuffer st = new StringBuffer();
          StringBuffer sb = new StringBuffer();
          for (int x = 0; x < c.length; x++){
           switch(c[x]){
           case '>':
            sb.append("&gt;");
            break;
           case '<':
            sb.append("&lt;");
            break;
           case '&':
            sb.append("&amp;");
            break;
```

```
                case '"':
                 sb.append("&quot;");
                 break;
                case '":
                 sb.append("&quot;");
                 break;
                default:
                 if (Character.isDefined(c[x]))
                  sb.append(c[x]);
                }
               }
               nvArr[z] = sb.toString();
               }
               return nvArr;
        }
        public void setEmail(int index, String inp){
            if (this.mEmail == null) {
               if (index < 0) {
                this.mEmail = new String[1];
                this.mEmail[0] = inp;
               } else {
                this.mEmail = new String[index + 1];
                this.mEmail[index] = inp;
               }
            } else if (index < 0) {
                           String [] newEmail = new String[this.mEmail.length + 1];
                           java.lang.System.arraycopy((Object)mEmail, 0, (Object)newEmail,
    0, this.mEmail.length);
                           newEmail[newEmail.length - 1] = inp;
                           mEmail = newEmail;
               } else if (index >= this.mEmail.length){
                           String [] newEmail = new String[index + 1];
                           java.lang.System.arraycopy((Object)mEmail, 0, (Object)newEmail,
    0, this.mEmail.length);
                           newEmail[index] = inp;
```

```
                    mEmail = newEmail;
          } else {
                    this.mEmail[index] = inp;
          }
}
public void setEmail(String [] inp){
          this.mEmail = inp;
}
public void emailFromXML(String n){
          setEmail(-1, n);
}
public String [] getInternet(){
  return mInternet;
}
public String getInternet(int index){
          if (this.mInternet == null)
            return null;
          if (index >= this.mInternet.length)
            return null;
          if (index < 0 && -index > this.mInternet.length)
            return null;
          if (index >= 0) return this.mInternet[index];
          return this.mInternet[this.mInternet.length + index];
}
public String [] getInternetToXML(){
          String [] valArr = getInternet();
          if (valArr == null) return null;
          String [] nvArr = new String[valArr.length];
          for (int z = 0; z < nvArr.length; z++){
          char [] c = valArr[z].toCharArray();
          StringBuffer st = new StringBuffer();
          StringBuffer sb = new StringBuffer();
          for (int x = 0; x < c.length; x++){
            switch(c[x]){
            case '>':
                sb.append("&gt;");
                break;
              case '<':
                sb.append("&lt;");
                break;
             case '&':
               sb.append("&amp;");
               break;
             case '"':
               sb.append("&quot;");
               break;
            case '\'':
              sb.append("&quot;");
              break;
            default:
              if (Character.isDefined(c[x]))
                sb.append(c[x]);
            }
          }
          nvArr[z] = sb.toString();
          }
          return nvArr;
}
public void setInternet(int index, String inp){
     if (this.mInternet == null) {
          if (index < 0) {
           this.mInternet = new String[1];
           this.mInternet[0] = inp;
          } else {
           this.mInternet = new String[index + 1];
           this.mInternet[index] = inp;
          }
    } else if (index < 0) {
                    String [] newInternet = new String[this.mInternet.length + 1];
```

```
                        java.lang.System.arraycopy((Object)mInternet, 0,
(Object)newInternet, 0, this.mInternet.length);
                        newInternet[newInternet.length - 1] = inp;
                        mInternet = newInternet;
                } else if (index >= this.mInternet.length){
                        String [] newInternet = new String[index + 1];
                        java.lang.System.arraycopy((Object)mInternet, 0,
(Object)newInternet, 0, this.mInternet.length);
                        newInternet[index] = inp;
                        mInternet = newInternet;
                } else {
                        this.mInternet[index] = inp;
                }
        }
        public void setInternet(String [] inp){
                this.mInternet = inp;
        }
        public void internetFromXML(String n){
                setInternet(-1, n);
        }
}

package com.veo.xdk.dev.schema.test.blib;

import com.veo.vsp.doclet.meta.Document;
public class Business extends Party {
        public static final String DOC_TYPE = "business";
        String aBusinessNumber;
        public Business(){
                super(DOC_TYPE);
        aBusinessNumber = new String();
        }
        public Business(String doc_type){
                super(doc_type);
        aBusinessNumber = new String();
        }
        static public Business initBusiness(String iBusinessNumber.String []
iPartyName,.AddressSet iAddressSet){
                Business obj = new Business();
                obj.initializeAll(iBusinessNumber, iPartyName, iAddressSet);
                return obj;
        }

        public void initializeAll(String iBusinessNumber,String [] iPartyName,AddressSet
iAddressSet){
                aBusinessNumber = iBusinessNumber;
                super.initializeAll(iPartyName, iAddressSet);
        }
        public String getBusinessNumber(){
          return aBusinessNumber;
        }
        public String getBusinessNumberToXML(){
                if (getBusinessNumber() == null) return null;
                char [] c = getBusinessNumber().toCharArray();
                StringBuffer sb = new StringBuffer();
                for (int x = 0; x < c.length; x++){
                 switch(c[x]){
                 case '>':
                  sb.append("&gt;");
                  break;
                 case '<':
                  sb.append("&lt;");
                  break;
                 case '&':
                  sb.append("&amp;");
                  break;
                 case '"':
                  sb.append("&quot;");
                  break;
                 case '\'':
                    sb.append("&quot;");
                    break;
                  default:
                   if (Character.isDefined(c[x]))
                     sb.append(c[x]);
                  }
                }
                return sb.toString();
        }
        public void setBusinessNumber(String inp){
                this.aBusinessNumber = inp;
        }
        public void businessNumberFromXML(String n){
                setBusinessNumber(n);
        }
}


import com.veo.vsp.doclet.meta.Document;
public class Party extends Document {
        public static final String DOC_TYPE = "party";
         String [] mPartyName;
         AddressSet mAddressSet;
        public Party(){
                super(DOC_TYPE);
        mPartyName = new String[0];
                this.mAddressSet = new AddressSet();
        }
        public Party(String doc_type){
                super(doc_type);
        mPartyName = new String[0];
                this.mAddressSet = new AddressSet();
        }
        static public Party initParty(String [] iPartyName,AddressSet iAddressSet){
                Party obj = new Party();
```

```
            obj.initializeAll(iPartyName, iAddressSet);
            return obj;
    }

    public void initializeAll(String [] iPartyName,AddressSet iAddressSet){
            mPartyName = iPartyName;
            mAddressSet = iAddressSet;
    }
    public String [] getPartyName(){
     return mPartyName;
    }
    public String getPartyName(int index){
            if (this.mPartyName == null)
             return null;
            if (index >= this.mPartyName.length)
              return null;
            if (index < 0 && -index > this.mPartyName.length)
              return null;
            if (index >= 0) return this.mPartyName[index];
            return this.mPartyName[this.mPartyName.length + index];
    }
    public String [] getPartyNameToXML(){
            String [] valArr = getPartyName();
            if (valArr == null) return null;
            String [] nvArr = new String[valArr.length];
            for (int z = 0; z < nvArr.length; z++){
            char [] c = valArr[z].toCharArray();
            StringBuffer st = new StringBuffer();
            StringBuffer sb = new StringBuffer();
            for (int x = 0; x < c.length; x++){
             switch(c[x]){
             case '>':
              sb.append("&gt;");
              break;
             case '<':
                    sb.append("&lt;");
                    break;
                  case '&':
                    sb.append("&amp;");
                    break;
                  case '"':
                    sb.append("&quot;");
                    break;
                  case '\'':
                    sb.append("&quot;");
                    break;
                  default:
                   if (Character.isDefined(c[x]))
                     sb.append(c[x]);
                  }
                }
                nvArr[z] = sb.toString();
                }
                return nvArr;
        }
        public void setPartyName(int index, String inp){
            if (this.mPartyName == null) {
                  if (index < 0) {
                   this.mPartyName = new String[1];
                   this.mPartyName[0] = inp;
                  } else {
                   this.mPartyName = new String[index + 1];
                   this.mPartyName[index] = inp;
                  }
            } else if (index < 0) {
                            String [] newPartyName = new String[this.mPartyName.length + 1];
                            java.lang.System.arraycopy((Object)mPartyName, 0,
(Object)newPartyName, 0, this.mPartyName.length);
                            newPartyName[newPartyName.length - 1] = inp;
                            mPartyName = newPartyName;
                } else if (index >= this.mPartyName.length){
                        String [] newPartyName = new String[index + 1];
                        java.lang.System.arraycopy((Object)mPartyName, 0,
(Object)newPartyName, 0, this.mPartyName.length);
                        newPartyName[index] = inp;
                        mPartyName = newPartyName;
                } else {
                        this.mPartyName[index] = inp;
                }
        }
        public void setPartyName(String [] inp){
                this.mPartyName = inp;
        }
        public void partyNameFromXML(String n){
                setPartyName(-1, n);
        }
        public AddressSet getAddressSet(){
         return mAddressSet;
        }
        public void setAddressSet(AddressSet inp){
                this.mAddressSet = inp;
        }
}

package com.veo.xdk.dev.schema.test.blib;

import com.veo.vsp.doclet.meta.Document;
public class Person extends Party {
        public static final String DOC_TYPE = "person";
         String aSSN;
        public Person(){
                super(DOC_TYPE);
        aSSN = null;
        }
        public Person(String doc_type){
                super(doc_type);
        aSSN = null;
        }
        static public Person initPerson(String iSSN,String [] iPartyName,AddressSet
iAddressSet){
                Person obj = new Person();
                obj.initializeAll(iSSN, iPartyName, iAddressSet);
                return obj;
        }

        public void initializeAll(String iSSN,String [] iPartyName,AddressSet iAddressSet){
                aSSN = iSSN;
                super.initializeAll(iPartyName, iAddressSet);
        }
        public String getSSN(){
          return aSSN;
        }
        public String getSSNToXML(){
                if (getSSN() == null) return null;
                char [] c = getSSN().toCharArray();
                StringBuffer sb = new StringBuffer();
                for (int x = 0; x < c.length; x++){
                  switch(c[x]){
                  case '>':
                    sb.append("&gt;");
                    break;
                  case '<':
                    sb.append("&lt;");
                    break;
                  case '&':
                    sb.append("&amp;");
                    break;
                  case '"':
                    sb.append("&quot;");
                    break;
                case '\"':
                 sb.append("&quot;");
                 break;
                default:
                 if (Character.isDefined(c[x]))
                  sb.append(c[x]);
                }
               }
               return sb.toString();
        }
        public void setSSN(String inp){
               this.aSSN = inp;
        }
        public void sSNFromXML(String n){
               setSSN(n);
        }
}

package com.veo.xdk.dev.schema.test.blib;

import com.veo.vsp.doclet.meta.Document;
public class PrototypeService extends Document {
        public static final String DOC_TYPE = "prototype.service";
         String mServiceName;
         String [] mServiceTerms;
         String [] mServiceLocation;
         ServiceOperation [] mServiceOperation;
        public PrototypeService(){
               super(DOC_TYPE);
        mServiceName = new String();
        mServiceTerms = new String[0];
        mServiceLocation = new String[0];
               this.mServiceOperation = new ServiceOperation[0];
        }
        public PrototypeService(String doc_type){
```

```
		super(doc_type);
	mServiceName = new String();
	mServiceTerms = new String[0];
	mServiceLocation = new String[0];
		this.mServiceOperation = new ServiceOperation[0];
	}
	static public PrototypeService initPrototypeService(String iServiceName,String []
iServiceTerms,String [] iServiceLocation,ServiceOperation [] iServiceOperation){
		PrototypeService obj = new PrototypeService();
		obj.initializeAll(iServiceName, iServiceTerms, iServiceLocation,
iServiceOperation);
		return obj;
	}

	public void initializeAll(String iServiceName,String [] iServiceTerms,String []
iServiceLocation,ServiceOperation [] iServiceOperation){
		mServiceName = iServiceName;
		mServiceTerms = iServiceTerms;
		mServiceLocation = iServiceLocation;
		mServiceOperation = iServiceOperation;
	}
	public String getServiceName(){
	 return mServiceName;
	}
	public String getServiceNameToXML(){
		if (getServiceName() == null) return null;
		char [] c = getServiceName().toCharArray();
		StringBuffer sb = new StringBuffer();
		for (int x = 0; x < c.length; x++){
		 switch(c[x]){
		 case '>':
		  sb.append("&gt;");
		  break;
		 case '<':
		  sb.append("&lt;");
```

```
                break;
              case '&':
                sb.append("&amp;");
                break;
              case '"':
                sb.append("&quot;");
                break;
              case '\'':
                sb.append("&quot;");
                break;
              default:
                if (Character.isDefined(c[x]))
                  sb.append(c[x]);
              }
            }
            return sb.toString();
    }
    public void setServiceName(String inp){
            this.mServiceName = inp;
    }
    public void serviceNameFromXML(String n){
            setServiceName(n);
    }
    public String [] getServiceTerms(){
     return mServiceTerms;
    }
    public String getServiceTerms(int index){
            if (this.mServiceTerms == null)
             return null;
            if (index >= this.mServiceTerms.length)
              return null;
            if (index < 0 && -index > this.mServiceTerms.length)
              return null;
            if (index >= 0) return this.mServiceTerms[index];
            return this.mServiceTerms[this.mServiceTerms.length + index];
```

```
}
public String [] getServiceTermsToXML(){
        String [] valArr = getServiceTerms();
        if (valArr == null) return null;
        String [] nvArr = new String[valArr.length];
        for (int z = 0; z < nvArr.length; z++){
        char [] c = valArr[z].toCharArray();
        StringBuffer st = new StringBuffer();
        StringBuffer sb = new StringBuffer();
        for (int x = 0; x < c.length; x++){
         switch(c[x]){
         case '>':
          sb.append("&gt;");
          break;
         case '<':
          sb.append("&lt;");
          break;
         case '&':
          sb.append("&amp;");
          break;
         case '"':
          sb.append("&quot;");
          break;
         case ''':
          sb.append("&quot;");
          break;
         default:
          if (Character.isDefined(c[x]))
           sb.append(c[x]);
         }
        }
        nvArr[z] = sb.toString();
        }
        return nvArr;
}
```

```
        public void setServiceTerms(int index, String inp){
            if (this.mServiceTerms == null) {
                if (index < 0) {
                  this.mServiceTerms = new String[1];
                  this.mServiceTerms[0] = inp;
                } else {
                  this.mServiceTerms = new String[index + 1];
                  this.mServiceTerms[index] = inp;
                }
            } else if (index < 0) {
                            String [] newServiceTerms = new String[this.mServiceTerms.length
+ 1];
                            java.lang.System.arraycopy((Object)mServiceTerms, 0,
(Object)newServiceTerms, 0, this.mServiceTerms.length);
                            newServiceTerms[newServiceTerms.length - 1] = inp;
                            mServiceTerms = newServiceTerms;
                } else if (index >= this.mServiceTerms.length){
                            String [] newServiceTerms = new String[index + 1];
                            java.lang.System.arraycopy((Object)mServiceTerms, 0,
(Object)newServiceTerms, 0, this.mServiceTerms.length);
                            newServiceTerms[index] = inp;
                            mServiceTerms = newServiceTerms;
                } else {
                            this.mServiceTerms[index] = inp;
                }
        }
        public void setServiceTerms(String [] inp){
                this.mServiceTerms = inp;
        }
        public void serviceTermsFromXML(String n){
                setServiceTerms(-1, n);
        }
        public String [] getServiceLocation(){
          return mServiceLocation;
        }
```

```
public String getServiceLocation(int index){
        if (this.mServiceLocation == null)
          return null;
        if (index >= this.mServiceLocation.length)
          return null;
        if (index < 0 && -index > this.mServiceLocation.length)
          return null;
        if (index >= 0) return this.mServiceLocation[index];
        return this.mServiceLocation[this.mServiceLocation.length + index];
}
public String [] getServiceLocationToXML(){
        String [] valArr = getServiceLocation();
        if (valArr == null) return null;
        String [] nvArr = new String[valArr.length];
        for (int z = 0; z < nvArr.length; z++){
        char [] c = valArr[z].toCharArray();
        StringBuffer st = new StringBuffer();
        StringBuffer sb = new StringBuffer();
        for (int x = 0; x < c.length; x++){
         switch(c[x]){
         case '>':
          sb.append("&gt;");
          break;
         case '<':
          sb.append("&lt;");
          break;
         case '&':
          sb.append("&amp;");
          break;
         case '"':
          sb.append("&quot;");
          break;
         case '\'':
          sb.append("&quot;");
          break;
```

```
                default:
                 if (Character.isDefined(c[x]))
                   sb.append(c[x]);
                }
              }
              nvArr[z] = sb.toString();
              }
              return nvArr;
         }
         public void setServiceLocation(int index, String inp){
             if (this.mServiceLocation == null) {
                 if (index < 0) {
                  this.mServiceLocation = new String[1];
                  this.mServiceLocation[0] = inp;
                 } else {
                  this.mServiceLocation = new String[index + 1];
                  this.mServiceLocation[index] = inp;
                 }
            } else if (index < 0) {
                          String [] newServiceLocation = new
String[this.mServiceLocation.length + 1];
                          java.lang.System.arraycopy((Object)mServiceLocation, 0,
(Object)newServiceLocation, 0, this.mServiceLocation.length);
                          newServiceLocation[newServiceLocation.length - 1] = inp;
                          mServiceLocation = newServiceLocation;
                 } else if (index >= this.mServiceLocation.length){
                          String [] newServiceLocation = new String[index + 1];
                          java.lang.System.arraycopy((Object)mServiceLocation, 0,
(Object)newServiceLocation, 0, this.mServiceLocation.length);
                          newServiceLocation[index] = inp;
                          mServiceLocation = newServiceLocation;
                 } else {
                          this.mServiceLocation[index] = inp;
                 }
         }
```

```
        public void setServiceLocation(String [] inp){
                this.mServiceLocation = inp;
        }
        public void serviceLocationFromXML(String n){
                setServiceLocation(-1, n);
        }
        public ServiceOperation [] getServiceOperation(){
          return mServiceOperation;
        }
        public ServiceOperation getServiceOperation(int index){
                if (this.mServiceOperation == null)
                  return null;
                if (index >= this.mServiceOperation.length)
                  return null;
                if (index < 0 && -index > this.mServiceOperation.length)
                  return null;
                if (index >= 0) return this.mServiceOperation[index];
                return this.mServiceOperation[this.mServiceOperation.length + index];
        }
        public void setServiceOperation(int index, ServiceOperation inp){
            if (this.mServiceOperation == null) {
                if (index < 0) {
                 this.mServiceOperation = new ServiceOperation[1];
                 this.mServiceOperation[0] = inp;
                } else {
                 this.mServiceOperation = new ServiceOperation[index + 1];
                 this.mServiceOperation[index] = inp;
                }
            } else if (index < 0) {
                        ServiceOperation [] newServiceOperation = new
ServiceOperation[this.mServiceOperation.length + 1];
                        java.lang.System.arraycopy((Object)mServiceOperation, 0,
(Object)newServiceOperation, 0, this.mServiceOperation.length);
                        newServiceOperation[newServiceOperation.length - 1] = inp;
                        mServiceOperation = newServiceOperation;
                } else if (index >= this.mServiceOperation.length){
                        ServiceOperation [] newServiceOperation = new
ServiceOperation[index + 1];
                        java.lang.System.arraycopy((Object)mServiceOperation, 0,
(Object)newServiceOperation, 0, this.mServiceOperation.length);
                        newServiceOperation[index] = inp;
                        mServiceOperation = newServiceOperation;
                } else {
                        this.mServiceOperation[index] = inp;
                }
        }
        public void setServiceOperation(ServiceOperation [] inp){
                this.mServiceOperation = inp;
        }
}

package com.veo.xdk.dev.schema.test.blib;

import com.veo.vsp.doclet.meta.Document;
public class Service extends Document {
        public static final String DOC_TYPE = "service";
        String mServiceName;
        String mServiceLocation;
        ServiceOperation [] mServiceOperation;
        String mServiceTerms;
        public Service(){
                super(DOC_TYPE);
        mServiceName = new String();
        mServiceLocation = new String();
                this.mServiceOperation = new ServiceOperation[0];
        mServiceTerms = new String();
        }
        public Service(String doc_type){
                super(doc_type);
        mServiceName = new String();
        mServiceLocation = new String();
                this.mServiceOperation = new ServiceOperation[0];
        mServiceTerms = new String();
        }
        static public Service initService(String iServiceName,String
iServiceLocation,ServiceOperation [] iServiceOperation,String iServiceTerms){
                Service obj = new Service();
                obj.initializeAll(iServiceName, iServiceLocation, iServiceOperation,
iServiceTerms);
                return obj;
        }

        public void initializeAll(String iServiceName,String
iServiceLocation,ServiceOperation [] iServiceOperation,String iServiceTerms){
                mServiceName = iServiceName;
                mServiceLocation = iServiceLocation;
                mServiceOperation = iServiceOperation;
                mServiceTerms = iServiceTerms;
        }
        public String getServiceName(){
         return mServiceName;
        }
        public String getServiceNameToXML(){
                if (getServiceName() == null) return null;
                char [] c = getServiceName().toCharArray();
                StringBuffer sb = new StringBuffer();
                for (int x = 0; x < c.length; x++){
                 switch(c[x]){
                 case '>':
                  sb.append("&gt;");
                  break;
                 case '<':
                  sb.append("&lt;");
                  break;
                 case '&':
```

```
                sb.append("&amp;");
                break;
              case '"':
                sb.append("&quot;");
                break;
              case '\'':
                sb.append("&quot;");
                break;
              default:
                if (Character.isDefined(c[x]))
                  sb.append(c[x]);
              }
            }
            return sb.toString();
    }
    public void setServiceName(String inp){
            this.mServiceName = inp;
    }
    public void serviceNameFromXML(String n){
            setServiceName(n);
    }
    public String getServiceLocation(){
      return mServiceLocation;
    }
    public String getServiceLocationToXML(){
            if (getServiceLocation() == null) return null;
            char [] c = getServiceLocation().toCharArray();
            StringBuffer sb = new StringBuffer();
            for (int x = 0; x < c.length; x++){
              switch(c[x]){
              case '>':
                sb.append("&gt;");
                break;
              case '<':
                sb.append("&lt;");
```

```
                  break;
                case '&':
                  sb.append("&amp;");
                  break;
                case '"':
                  sb.append("&quot;");
                  break;
                case '\'':
                  sb.append("&quot;");
                  break;
                default:
                  if (Character.isDefined(c[x]))
                    sb.append(c[x]);
                }
            }
            return sb.toString();
    }
    public void setServiceLocation(String inp){
            this.mServiceLocation = inp;
    }
    public void serviceLocationFromXML(String n){
            setServiceLocation(n);
    }
    public ServiceOperation [] getServiceOperation(){
      return mServiceOperation;
    }
    public ServiceOperation getServiceOperation(int index){
            if (this.mServiceOperation == null)
              return null;
            if (index >= this.mServiceOperation.length)
              return null;
            if (index < 0 && -index > this.mServiceOperation.length)
              return null;
            if (index >= 0) return this.mServiceOperation[index];
            return this.mServiceOperation[this.mServiceOperation.length + index];
```

```
        }
        public void setServiceOperation(int index, ServiceOperation inp){
            if (this.mServiceOperation == null) {
                if (index < 0) {
                 this.mServiceOperation = new ServiceOperation[1];
                 this.mServiceOperation[0] = inp;
                } else {
                 this.mServiceOperation = new ServiceOperation[index + 1];
                 this.mServiceOperation[index] = inp;
                }
            } else if (index < 0) {
                        ServiceOperation [] newServiceOperation = new
ServiceOperation[this.mServiceOperation.length + 1];
                        java.lang.System.arraycopy((Object)mServiceOperation. 0,
(Object)newServiceOperation. 0, this.mServiceOperation.length);
                        newServiceOperation[newServiceOperation.length - 1] = inp;
                        mServiceOperation = newServiceOperation;
                } else if (index >= this.mServiceOperation.length){
                        ServiceOperation [] newServiceOperation = new
ServiceOperation[index + 1];
                        java.lang.System.arraycopy((Object)mServiceOperation, 0,
(Object)newServiceOperation, 0, this.mServiceOperation.length);
                        newServiceOperation[index] = inp;
                        mServiceOperation = newServiceOperation;
                } else {
                        this.mServiceOperation[index] = inp;
                }
        }
        public void setServiceOperation(ServiceOperation [] inp){
                this.mServiceOperation = inp;
        }
        public String getServiceTerms(){
         return mServiceTerms;
        }
        public String getServiceTermsToXML(){
                if (getServiceTerms() == null) return null;
                char [] c = getServiceTerms().toCharArray();
                StringBuffer sb = new StringBuffer();
                for (int x = 0; x < c.length; x++){
                 switch(c[x]){
                 case '>':
                  sb.append("&gt;");
                  break;
                 case '<':
                  sb.append("&lt;");
                  break;
                 case '&':
                  sb.append("&amp;");
                  break;
                 case '"':
                  sb.append("&quot;");
                  break;
                 case '\'':
                  sb.append("&quot;");
                  break;
                default:
                 if (Character.isDefined(c[x]))
                  sb.append(c[x]);
                }
               }
               return sb.toString();
        }
        public void setServiceTerms(String inp){
               this.mServiceTerms = inp;
        }
        public void serviceTermsFromXML(String n){
               setServiceTerms(n);
        }
}
```

```
package com.veo.xdk.dev.schema.test.blib;

import com.veo.vsp.doclet.meta.Document;
public class ServiceOperation extends Document {
        public static final String DOC_TYPE = "service.operation";
         String mServiceOperationName;
         String mServiceOperationLocation;
         String mServiceOperationInput;
         String mServiceOperationOutput;
        public ServiceOperation(){
                super(DOC_TYPE);
        mServiceOperationName = new String();
        mServiceOperationLocation = new String();
        mServiceOperationInput = new String();
        mServiceOperationOutput = new String();
        }
        public ServiceOperation(String doc_type){
                super(doc_type);
        mServiceOperationName = new String();
        mServiceOperationLocation = new String();
        mServiceOperationInput = new String();
        mServiceOperationOutput = new String();
        }
        static public ServiceOperation initServiceOperation(String
iServiceOperationName,String iServiceOperationLocation.String iServiceOperationInput,String
iServiceOperationOutput){
                ServiceOperation obj = new ServiceOperation();
                obj.initializeAll(iServiceOperationName, iServiceOperationLocation,
iServiceOperationInput, iServiceOperationOutput);
                return obj;
        }

        public void InitializeAll(String iServiceOperationName,String
iServiceOperationLocation,String iServiceOperationInput,String iServiceOperationOutput){
                mServiceOperationName = iServiceOperationName;
```

```
            mServiceOperationLocation = iServiceOperationLocation;
            mServiceOperationInput = iServiceOperationInput;
            mServiceOperationOutput = iServiceOperationOutput;
    }
    public String getServiceOperationName(){
     return mServiceOperationName;
    }
    public String getServiceOperationNameToXML(){
            if (getServiceOperationName() == null) return null;
            char [] c = getServiceOperationName().toCharArray();
            StringBuffer sb = new StringBuffer();
            for (int x = 0; x < c.length; x++){
             switch(c[x]){
             case '>':
              sb.append("&gt;");
              break;
             case '<':
              sb.append("&lt;");
              break;
             case '&':
              sb.append("&amp;");
              break;
             case '"':
              sb.append("&quot;");
              break;
             case '\'':
              sb.append("&quot;");
              break;
             default:
              if (Character.isDefined(c[x]))
               sb.append(c[x]);
             }
            }
           return sb.toString();
   }
```

```
public void setServiceOperationName(String inp){
        this.mServiceOperationName = inp;
}
public void serviceOperationNameFromXML(String n){
        setServiceOperationName(n);
}
public String getServiceOperationLocation(){
 return mServiceOperationLocation;
}
public String getServiceOperationLocationToXML(){
        if (getServiceOperationLocation() == null) return null;
        char [] c = getServiceOperationLocation().toCharArray();
        StringBuffer sb = new StringBuffer();
        for (int x = 0; x < c.length; x++){
         switch(c[x]){
         case '>':
          sb.append("&gt;");
          break;
         case '<':
          sb.append("&lt;");
          break;
        case '&':
          sb.append("&amp;");
          break;
        case '"':
          sb.append("&quot;");
          break;
        case '\'':
          sb.append("&quot:");
          break;
        default:
         if (Character.isDefined(c[x]))
          sb.append(c[x]);
        }
       }
```

```
            return sb.toString();
  }
  public void setServiceOperationLocation(String inp){
            this.mServiceOperationLocation = inp;
  }
  public void serviceOperationLocationFromXML(String n){
            setServiceOperationLocation(n);
  }
  public String getServiceOperationInput(){
   return mServiceOperationInput;
  }
  public String getServiceOperationInputToXML(){
            if(getServiceOperationInput() == null) return null;
            char [] c = getServiceOperationInput().toCharArray();
            StringBuffer sb = new StringBuffer();
            for (int x = 0; x < c.length; x++){
             switch(c[x]){
             case '>':
              sb.append("&gt;");
              break;
             case '<':
              sb.append("&lt;");
              break;
             case '&':
              sb.append("&amp;");
              break;
             case '"':
              sb.append("&quot;");
              break;
             case '\'':
              sb.append("&quot;");
              break;
             default:
              if(Character.isDefined(c[x]))
               sb.append(c[x]);
            }
        }
        return sb.toString();
    }
    public void setServiceOperationInput(String inp){
            this.mServiceOperationInput = inp;
    }
    public void serviceOperationInputFromXML(String n){
            setServiceOperationInput(n);
    }
    public String getServiceOperationOutput(){
     return mServiceOperationOutput;
    }
    public String getServiceOperationOutputToXML(){
            if (getServiceOperationOutput() == null) return null;
            char [] c = getServiceOperationOutput().toCharArray();
            StringBuffer sb = new StringBuffer();
            for (int x = 0; x < c.length; x++){
             switch(c[x]){
             case '>':
              sb.append("&gt;");
              break;
             case '<':
              sb.append("&lt;");
              break;
            case '&':
              sb.append("&amp;");
              break;
            case '"':
             sb.append("&quot;");
             break;
            case '\'':
             sb.append("&quot;");
             break;
            default:
                    if (Character.isDefined(c[x]))
                      sb.append(c[x]);
                   }
                 }
                 return sb.toString();
        }
        public void setServiceOperationOutput(String inp){
                 this.mServiceOperationOutput = inp;
        }
        public void serviceOperationOutputFromXML(String n){
                 setServiceOperationOutput(n);
        }
}

package com.veo.xdk.dev.schema.test.blib;

import com.veo.vsp.doclet.meta.Document;
public class ServiceSet extends Document {
        public static final String DOC_TYPE = "service.set";
         Service [] mService;
        public ServiceSet(){
                 super(DOC_TYPE);
                 this.mService = new Service[0];
        }
        public ServiceSet(String doc_type){
                 super(doc_type);
                 this.mService = new Service[0];
        }
        static public ServiceSet initServiceSet(Service [] iService){
                 ServiceSet obj = new ServiceSet();
                 obj.initializeAll(iService);
                 return obj;
        }

        public void initializeAll(Service [] iService){
```

```
                        mService = iService;
                 }
                 public Service [] getService(){
                   return mService;
                 }
                 public Service getService(int index){
                        if (this.mService == null)
                         return null;
                        if (index >= this.mService.length)
                          return null;
                        if (index < 0 && -index > this.mService.length)
                          return null;
                        if (index >= 0) return this.mService[index];
                        return this.mService[this.mService.length + index];
                }
                public void setService(int index, Service inp){
                    if (this.mService == null) {
                        if (index < 0) {
                         this.mService = new Service[1];
                         this.mService[0] = inp;
                        } else {
                         this.mService = new Service[index + 1];
                         this.mService[index] = inp;
                        }
                 } else if (index < 0) {
                              Service [] newService = new Service[this.mService.length + 1];
                              java.lang.System.arraycopy((Object)mService, 0,
          (Object)newService, 0, this.mService.length);
                              newService[newService.length - 1] = inp;
                              mService = newService;
                      } else if (index >= this.mService.length){
                              Service [] newService = new Service[index + 1];
                              java.lang.System.arraycopy((Object)mService, 0,
          (Object)newService, 0, this.mService.length);
                              newService[index] = inp;
                        mServiсe = newService;
                } else {
                        this.mService[index] = inp;
                }
        }
        public void setService(Service [] inp){
                this.mService = inp;
        }
}
```

In addition to the above-mentioned JAVA flowbean, a conversion code is made in order to translate into XML and JAVA from XML from JAVA as shown below.

Java to XML

```
<!DOCTYPE tree SYSTEM "tree.dtd">
<tree source = "null" pass-through = "false">
<before>
 <vardef name = "attribute.def">
  <element source = "ATTRIBUTE" class = "NAME" type = "5" position = "-2">
   <parse>
    <data class = "java.lang.String" position = "-2"/>
   </parse>
  </element>
</vardef>
<verdef name = "pcdata.def">
 <element source = "PCDATA" class = "NAME" type = "4" position = "-2">
  <parse>
   <data class = "999" type = "6" position = "-2"/>
  </parse>
 </element>
</vardef>
<vardef name = "content.def">
 <element source = "PCDATA">
```

```
        <parse>
          <data class = "999" type = "6" position = "-2"/>
        </parse>
      </element>
    </vardef>
    <vardef name = "ServiceSet.var">
    <element source = "com.veo.xdk.dev.schema.test.blib.ServiceSet" class = "service.set" type =
   "4" position = "-2">
      <parse>
        <callvar name = "Service.var"/>
      </parse>
    </element>
    </vardef>
    <vardef name = "PrototypeService.var">
    <element source = "com.veo.xdk.dev.schema.test.blib.PrototypeService" class =
   "prototype.service" type = "4" position = "-2">
      <parse>
        <callvar name = "pcdata.def" parms = "setSource ServiceNameToXML setGenerator
   service.name"/>
        <callvar name = "pcdata.def" parms = "setSource ServiceTermsToXML setGenerator
   service.terms"/>
        <callvar name = "pcdata.def" parms = "setSource ServiceLocationToXML setGenerator
   service.location"/>
        <callvar name = "ServiceOperation.var"/>
      </parse>
    </element>
    </vardef>
   <vardef name = "Service.var">
   <element source = "com.veo.xdk.dev.schema.test.blib.Service" class = "service" type = "8"
  position = "0">
     <parse>
       <callvar name = "pcdata.def" parms = "setSource ServiceNameToXML setGenerator
  service.name"/>
       <callvar name = "pcdata.def" parms = "setSource ServiceLocationToXML setGenerator
  service.location"/>
```

```
        <callvar name = "ServiceOperation.var"/>
        <callvar name = "pcdata.def" parms = "setSource ServiceTermsToXML setGenerator
  service.terms"/>
      </parse>
    </element>
    </vardef>
    <vardef name = "ServiceOperation.var">
    <element source = "com.veo.xdk.dev.schema.test.blib.ServiceOperation" class =
  "service.operation" type = "4" position = "-2">
      <parse>
        <callvar name = "pcdata.def" parms = "setSource ServiceOperationNameToXML
  setGenerator service.operation.name"/>
        <callvar name = "pcdata.def" parms = "setSource ServiceOperationLocationToXML
  setGenerator service.operation.location"/>
        <callvar name = "pcdata.def" parms = "setSource ServiceOperationInputToXML
  setGenerator service.operation.input"/>
        <callvar name = "pcdata.def" parms = "setSource ServiceOperationOutputToXML
  setGenerator service.operation.output"/>
      </parse>
    </element>
    </vardef>
  </before>
  <parse>
  <callvar name = "ServiceSet.var"/>
  <callvar name = "PrototypeService.var"/>
  <callvar name = "Service.var"/>
  <callvar name = "ServiceOperation.var"/>
  </parse>
  </tree>
```

XML to Java

```
<!DOCTYPE tree SYSTEM "tree.dtd">
<tree source = "null" pass-through = "false">
<before>
```

```
 <vardef name = "business.var">
  <element source = "business"
       class = "com.veo.xdk.dev.schema.test.blib.Business"
       type = "7" setter = "setBusiness">
   <before>
    <onattribute name = "business.number">
     <actions>
      <callmeth name = "businessNumberFromXML">
       <parms>
        <getattr name = "business.number"/>
       </parms>
      </callmeth>
     </actions>
    </onattribute>
   </before>
   <parse>
    <callvar name = "party.name.var" parms = "setPosition -1"/>
    <callvar name = "address.set.var"/>
   </parse>
  </element>
</vardef>
<vardef name = "party.name.var">
 <element source = "party.name" setter = "partyNameFromXML" position = "-1" class =
"java.lang.String">
  <parse>
     <data class = "java.lang.String" position = "0"/>
  </parse>
 </element>
</vardef>
<vardef name = "city.var">
 <element source = "city" setter = "cityFromXML" position = "-1" class = "java.lang.String">
  <parse>
     <data class = "java.lang.String" position = "0"/>
  </parse>
 </element>
 </vardef>
 <vardef name = "internet.var">
  <element source = "internet" setter = "internetFromXML" position = "-1" class =
"java.lang.String">
   <parse>
     <data class = "java.lang.String" position = "0"/>
   </parse>
  </element>
 </vardef>
 <vardef name = "country.var">
  <element source = "country" setter = "countryFromXML" position = "-1" class =
"java.lang.String">
   <parse>
     <data class = "java.lang.String" position = "0"/>
   </parse>
  </element>
 </vardef>
 <vardef name = "state.var">
  <element source = "state" setter = "stateFromXML" position = "-1" class = "java.lang.String">
   <parse>
     <data class = "java.lang.String" position = "0"/>
   </parse>
  </element>
 </vardef>
 <vardef name = "email.var">
  <element source = "email" setter = "emailFromXML" position = "-1" class =
"java.lang.String">
   <parse>
     <data class = "java.lang.String" position = "0"/>
   </parse>
  </element>
 </vardef>
 <vardef name = "address.physical.var">
  <element source = "address.physical"
       class = "com.veo.xdk.dev.schema.test.blib.AddressPhysical"
```

```
            type = "7" setter = "setAddressPhysical">
        <before>
        </before>
        <parse>
          <callvar name = "street.var" parms = "setPosition -1"/>
          <callvar name = "city.var" parms = "setPosition -1"/>
          <callvar name = "state.var" parms = "setPosition -1"/>
          <callvar name = "postcode.var" parms = "setPosition -1"/>
          <callvar name = "country.var" parms = "setPosition -1"/>
        </parse>
      </element>
    </vardef>
    <vardef name = "telephone.var">
      <element source = "telephone" setter = "telephoneFromXML" position = "-1" class =
"java.lang.String">
        <parse>
          <data class = "java.lang.String" position = "0"/>
        </parse>
      </element>
    </vardef>
    <vardef name = "person.var">
      <element source = "person"
            class = "com.veo.xdk.dev.schema.test.blib.Person"
            type = "7" setter = "setPerson">
        <before>
          <onattribute name = "SSN">
            <actions>
              <callmeth name = "sSNFromXML">
                <parms>
                  <getattr name = "SSN"/>
                </parms>
              </callmeth>
            </actions>
          </onattribute>
        </before>
```

```
   <parse>
    <callvar name = "party.name.var" parms = "setPosition -1"/>
    <callvar name = "address.set.var"/>
   </parse>
  </element>
 </vardef>
 <vardef name = "fax.var">
  <element source = "fax" setter = "faxFromXML" position = "-1" class = "java.lang.String">
   <parse>
     <data class = "java.lang.String" position = "0"/>
   </parse>
  </element>
 </vardef>
 <vardef name = "street.var">
  <element source = "street" setter = "streetFromXML" position = "-1" class =
"java.lang.String">
   <parse>
     <data class = "java.lang.String" position = "0"/>
   </parse>
  </element>
 </vardef>
 <vardef name = "address.set.var">
  <element source = "address.set"
      class = "com.veo.xdk.dev.schema.test.blib.AddressSet"
      type = "7" setter = "setAddressSet">
   <before>
   </before>
   <parse>
    <callvar name = "address.physical.var"/>
    <callvar name = "telephone.var" parms = "setPosition -1"/>
    <callvar name = "fax.var" parms = "setPosition -1"/>
    <callvar name = "email.var" parms = "setPosition -1"/>
    <callvar name = "internet.var" parms = "setPosition -1"/>
   </parse>
  </element>
 </vardef>
 <vardef name = "postcode.var">
  <element source = "postcode" setter = "postcodeFromXML" position = "-1" class =
 "java.lang.String">
    <parse>
      <data class = "java.lang.String" position = "0"/>
    </parse>
  </element>
 </vardef>
 <vardef name = "market.participant.var">
  <element source = "market.participant"
       class = "com.veo.xdk.dev.schema.test.blib.MarketParticipant"
       type = "7" position = "0">
   <before>
   </before>
   <parse>
    <callvar name = "business.var"/>
    <callvar name = "person.var"/>
   </parse>
  </element>
</vardef>
</before>
<parse>
<callvar name = "business.var"/>
<callvar name = "party.name.var"/>
<callvar name = "city.var"/>
<callvar name = "internet.var"/>
<callvar name = "country.var"/>
<callvar name = "state.var"/>
<callvar name = "email.var"/>
<callvar name = "address.physical.var"/>
<callvar name = "telephone.var"/>
<callvar name = "person.var"/>
<callvar name = "fax.var"/>
<callvar name = "street.var"/>
```

```
<callvar name = "address.set.var"/>
<callvar name = "postcode.var"/>
<callvar name = "market.participant.var"/>
</parse>
</tree>
```

Makefiles:

```
#
# this makefile was generated by bic version 0.0. 05/02/1998
#
#
#
#

# get the package name from the package argument passed to SchemaGen
PACKAGE_NAME = com/veo/xdk/dev/schema/test/blib

JAVA_SOURCES    +=        \
        MarketParticipant.java \
        Business.java \
        Person.java \
        Party.java \
        AddressPhysical.java \
        AddressSet.java \

MAKEFILE_MASTER_DIR = xxx
include $(MAKEFILE_MASTER_DIR)/Makefile.master

all:: $(JAVA_CLASSES)

#
# this makefile was generated by bic version 0.0. 05/02/1998
#
```

```
#
#
#

# get the package name from the package argument passed to SchemaGen
PACKAGE_NAME = com/veo/xdk/dev/schema/test/blib

JAVA_SOURCES    +=          \
          ServiceSet.java \
          PrototypeService.java \
          Service.java \
          ServiceOperation.java \

MAKEFILE_MASTER_DIR = xxx
include $(MAKEFILE_MASTER_DIR)/Makefile.master

all:: $(JAVA_CLASSES)
```

Finally, the XML document instances generated at run time according to the model above for one example follows:

```
<!DOCTYPE market.participant SYSTEM "market.participant.dtd" >
<market.participant>

<business business.number="1234567890" >

<party.name>IBM</party.name>

<address.set>
<address.physical>

<street>1 IBM Way</street>
<city>Palo Alto</city>
<state>CA</state>
```

```
<postcode>94304</postcode>
<country>USA</country>
</address.physical>

<telephone>123 456-7890</telephone>
<fax>123 456 0987</fax>
<email>ibmec@ibm.com</email>
</address.set>

</business>

</market.participant>

<!DOCTYPE service SYSTEM "service.dtd" >
<service.set>
<service>
<service.name>Order Service</service.name>
<service.location>www.ibm.com/order</service.location>

<service.operation>
<service.operation.name>Submit Order</service.operation.name>
<service.operation.location>www.ibm.com/order/submit</service.location>
<service.operation.input>urn:x-ibm:services:order:operations:po.dtd</service.operation.input>
<service.operation.output>urn:x-
ibm:services:order:operations:poack.dtd</service.operation.output>
</service.operation>

<service.operation>
<service.operation.name>Track Order</service.operation.name>
<service.operation.location>www.ibm.com/order/track</service.location>
<service.operation.input>urn:x-
ibm:services:order:operations:track.irequest.dtd</service.operation.input>
<service.operation.output>urn:x-
ibm:services:order:operations:track.iresponse.dtd</service.operation.output>
</service.operation>
```

```
</service>
(/service.set>
```

[0101]
Use a tool together with the BID photo composing machine (composer) application which provides the YUSA interface which edits a drug, drops, and a form, and a developer, The definition of the effective business interface which could make the definition of the business interface and was formed good is producible with the gestalt of an XML document. Thus, the run time instance of an example is a definition of the business interface for placing an order for the service for IBM used with the Ingram micro (Ingram Micro), in order to order a laptop computer from IBM. (There is no relation between an applicant, and IBM or the Ingram micro.) If the process of these is used, the user can use the document defined according to this invention, and can build the system which can do the program of a business interface.
[0102]
He could understand further CBL of this invention in XML/JAVA environment, and the role of the BID processor by explanation of the following of processing of a purchase order.
[0103]
The company A defines the purchase order document which uses vision program environment including the library of CBL DTD and a module, all are defined using a common business language element, and they include a data type and other translation information. PO of the company A may contain myna customisation to general transaction document specification, or may be made from the ground from a CBL module for an address, data and time, current, etc. rather than attached to the CBL library.
[0104]
(It is like the above-mentioned transact.dtd) CBL specification is made from a module and the document of general transaction document specification typifies the method connected with other CBL DTD.
[0105]
A compiler takes the definition of a purchase order and generates some different target forms. All such target forms can be acquired via "tree two tree (tree to tree)" modification of the original specification. JAVA flowbean (Bean) (a JAVA class, an argument, a data type, a method, and exception structure are made corresponding to the information on the schema definition of a purchase order.) in which it takes for this example and the most important one includes the data structure for the XML DTD (b) purchase order for the (a) purchase order (c) The purchase order according to purchase order DTD is changed into purchase order JAVA flowbean, Or they are loaded to a database, Or a data value is pulled out from the alignment program (d) purchase order JAVA flowbean which makes HTML (or collection of

XSL styles) in order to display a purchase order on a browser, Program which changes them into the XML document according to purchase order DTD and which does not align If it returns to a scenario now, the application of purchase will generate a purchase order and this purchase order will follow DTD specified as a service interface for the supplier who accepts a purchase order.
[0106]
A purser uses purchase order DTD and decomposes the purchase order instance to the information stream about the attribute value which an element and it include. Then, these property sets are changed into a JAVA event object corresponding by piling them up with the JAVA code. As a matter of fact, this conversion processes the portion of the XML document which put the seal as a command of the customer program language with which grammar is defined by DTD. These JAVA events are processed by the application which was generated by the compiler and which has aligned, and can load the data structure of JAVA flowbean.
[0107]
Since unlike the normal model in which changing an XML document into the one-set event for JAVA application, and processing it analyzes in-house-data structure and processing are not started until analysis is completed, a purser output is maintained. Since processing of the event base according to a BID definition will start the document application process of generating simultaneously immediately if the first event is emitted, it is a key which enables the function which was dramatically excellent in the processor.
[0108]
The JAVA program asked to an event various type is formed from the schema definition of those events. These listeners may be the programs made in order to perform business logic relevant to the XML definition of CBL, for example, may be codes which check a zip code by checking a database in relation to an address element. By registering with a document router, these listeners subscribe for an event and assign the event relevant to all the applicants who showed them interest.
[0109]
This architecture means that new listeners program and that announced officially and subscribed can be added without the knowledge of an existing thing, without affecting it. Each listeners have the cue which a router assigns the event, and two or more listeners can deal with an event in parallel at the pace of these selves.
[0110]
For the purchase order of the illustration here, - it is tied to an order input program -- I will come out -- a purchase order and stock are checked -- I will come out -- other services for the validity of a product description document and Fed Ex, or delivery can be checked -- I will come out -- address information - (since it is worth of a credit loan, or in order to provide a

public notice) Or they may be the listeners for the information of the buyer who could check the history of the order for the same processing based on getting to know who a customer is.
[0111]
The listeners who compounded can be made as shape of an early thing. (For example, the listeners of a purchase order may refer to it including these listeners here, or they may be referred to for themselves [these].)
[0112]
Drawing 11 shows the node of the market marker in the network of drawing 1. A market marker's node has the host 1103 to the document and document translator to the network interface 1101, the document purser 1102, and a host, and the front end 1104 including the basic structure of the system of drawing 3, In this example, it is called the router. The market marker module 1105 in this example An one-set business interface definition, Or it has other sufficient identifiers to support a market marker's function for the participant in a market, the CBL repository, and the compiler that fills all demands of the participant in a market. The router 1104 is provided with a participant's registry and document filter, answer the event generated by the purser with the output of the translator, and by a participant's registry. And the course of an input document is defined with the element and attribute filter between [to an XML event generation machine] listeners. Therefore, the fixed participant in a market may register and may receive the document suitable for the parameter specified beforehand. For example, the input document by specific DTD can be used for a filter document with the router 1104 including attributes, such as the number of purchase products beyond a threshold, or the maximum price of the document demand purchased. Then, a document which agrees to the information registered into the participant's registry with the router 1104 is sent to the registered participant.
[0113]
The router 1104 fills the demand of the local host services 1105 and 1106 again, for example, operates like the participant in a market like a market marker. Usually, the document received by the router 1104 is traversed, an address is determined, and such a document is sent, and again, via the translator 1103, if required, it will be sent to each address from the network interface 1101 there.
[0114]
A market marker is a server tied up together with business service of an one-set inside and the exterior, and makes the plan or trading community of imagination. Suitable service is referred to a server analyzing an input document, for example, passing the request for product data to a catalog server, or by sending a purchase order to an ERP system. A server deals with a translation task again and maps the information from the XML document of a company in the data format demanded by the document form used by the trading partner and its legacy

system.
[0115]
When a company presents a purchase order about the definition of the above-mentioned service, the XML parser of a server uses purchase order DTD, and changes a purchase order instance into the stream of an information event. Then, it is sent to the application programmed in order that these events might deal with the event of a predetermined type, and, in some cases, information is thoroughly sent to different business via the Internet. In the example of a purchase order, some applications may operate using the information which comes from a purser.
- An order input program processes purchase as a perfect message.
- An ERP system checks the stock for the product described by the purchase order.
- A customer database proves or updates a customer's address.
- An express company uses address information and schedules conveyance.
- A bank uses a credit card and permits a transaction.
[0116]
The trading partner should just agree about the structure, the contents, and an order of the business document which they exchange, and does not need to agree about the details of API. It is dependent on the business which provides service strictly how a document is processed and what kind of action result arises. This raises the integration to a business level from a system level. It makes it possible to provide the business partner with the interface in which business is clear in spite of the change in fulfillment of the internal art, systematization, or a process, and stable.
[0117]
Drawing 12, and 13 and 14 show the process performed in the market maker node of the system of drawing 11. In drawing 12, an input document is received in the network interface from a dispatch participant node (Step 1200). A document is parsed (Step 1201). A document is changed into a host's format from XML to JAVA, for example (Step 1202). The event and object which were host-format-ized are sent to router service after that (Step 1203). The service remembered to accept a document according to the type of a document and the contents of the document is identified (Step 1204). A part of document and document are sent to the identified service (Step 1205). Service is performed according to the contents of the document (Step 1206). The output data of service is generated (Step 1207). An output is changed into a document format from a JAVA format to an XML format, for example (Step 1208). Finally, an output document is sent to a participant node (Step 1209).
[0118]
Registration service is one of such the functions managed by a router. Therefore, a market participant document is accepted in the network interface shown in drawing 13 (Step 1300). A

market participant document is memorized by the business interface definition repository about a market maker node (Step 1301). In addition, a document is parsed (Step 1302). The parsed document is changed into a host's format (Step 1303). Next, a document is sent to router service (Step 1304). Router service contains the listeners who identify registration service according to the type of a document, and the contents of the document as it is an address of a document (Step 1305). The element of a document and a document is sent to registration service (Step 1306). RITORIBU [the service specifications needed] (Step 1307) according to a business interface definition in registration service. In Step 1308, when service specifications are collected, a router service filter is set according to a business interface definition and service specifications (Step 1309). Registration acknowledgement data is generated (Step 1310). Registration acknowledgement data is changed into a document format (Step 1311). Finally, an acknowledgement document notifies the participant by whom was seen off in the participant node and the register was done to the success reverse side by the market maker (Step 1312).

[0119]

The process in Step 1307 which collects the service specifications needed is shown about one example of drawing 14. This process starts by discovering the service business interface definition supported by the market participant (Step 1400). RITORIBU [a service definition / E mail transaction or web access for example, to a repository node] (Step 1401). Service specifications are memorized by the BID repository (Step 1402). A service business interface definition document is parsed (Step 1403). The parsed document is changed into a host's format (Step 1404). Next, a host's object is sent to router service (Step 1405). Registration service is identified according to the type and the contents of the document (Step 1406). Finally, the information on a service business interface definition document is sent to registration service (Step 1407), and is used according to the process of drawing 13.

[0120]

Drawing 15 shows the processor and component which process the data inputted into a market maker node according to this invention, and the sequence. A market maker node has the communication agent 1500 in a network interface. A communication agent is connected with XML parser 1501, and XML parser 1501 supplies an event to the XML processor 1502. The XML processor 1502 supplies an event to a document router. Feeding a document router in the document service 1504, the document service 1504 serves as an interface which supplies the received document to the enterprise SORUSHON software 1505 of a host system. The communication agent 1500 is the Internet interface which has a suitable protocol stack which supports HTTP, SMTP, FTP, or a protocol like other protocols. Therefore, the data to input may be inputted by XML syntax, the ASCII data syntax method, or other syntax so that a specific communication channel may be suited. All the documents received by non-XML

syntax are changed into XML, and are sent to an XML parser. In order to support the conversion to XML form [XML form / non-], the translation table 1506 is used.
[0121]
The changed document is supplied to the purser 1501. An XML parser parses the received XML document according to the document type definition which matches it. When an error is found, a purser returns a document to the communication agent 1500. Business interface definition compiler BIDC1507 works as a compiler for business interface definition data. The conversion rule which changes JAVA BINZU and the DTD file corresponding to a DTD file and a DTD file for an XML parser into JAVA BINZU is made by compiling BID data. An XML instance is changed into a JAVA instance by referring to these tools. Then, the BID compiler 1507 memorizes the DTD document 1508, and generates the JAVA document corresponding to 1509. An XML document is sent to the processor 1502 and the processor 1502 changes the document into a JAVA format. In a desirable system, the JAVA document which has the same status as the document type definition received in the XML format is generated. JAVA BINZU is sent to the document router 1503. The event listeners architecture which the document router 1503 received JAVA BINZU, and used the registry program, for example, mentioned the received class above is used, and it sends to suitable document service. The document service 1504 which received the document from the router 1503 in the form of JAVA BINZU is committed as ene TAPURAIZUSO Roussillon software. This has the service manager 1511 who manages routing to the registry service 1510 combined with the data flow which inputs the listeners of an XML event, and suitable service of the document to input. The document service manager 1511 performs management of registry service, maintenance of the consistency of a document, etc.
[0122]
Document service uses arbitrary pro PURAETARI APIs, or communicates with a back-end system using a more common form like a CORBA/COM interface and the other architecture.
[0123]
Drawing 16 is a heuristics diagram of a market maker and market participant structure concerning this invention. Therefore, the e-commerce transaction market concerning this invention can be logically systematized as shown in drawing 16. The market maker node 1600 is set up at the topmost part of the organization. A market maker node contains the resource which sets up the marketplace 1601. Such a resource includes market registry service etc. The register of the business 1602 is carried out to a marketplace by carrying out Publish of the business interface definition. A business interface definition defines the service 1603 about the commercial transaction in which business participates. The transaction 1604 and the service 1603 define an input and an output using the document 1605, and carry out outline of the relation on the commerce between the participants in a transaction. A document has the

content 1606 including the details of each transaction. A content depends for the method processed by the participant in a market, and the market maker on the document based on the e-commerce transaction network founded according to this invention thoroughly. The structure which is rich in robustness, extendibility, and intuitive ability so that an e-commerce transaction may be made possible on a communication network as a whole is provided.

[0124]

Therefore, in the system of illustration, this invention provides the plat form based on an XML processor, and the XML document is being used for it as an interface between the business systems by which loose coupling was carried out. A document is transmitted between business, and before it goes into the business system of a company, it is processed by the specific node. Therefore, a plat form makes e-commerce application between business possible by specifying the business document and form of a common group, when the internal commerce plat form and processing in which each business systems differ, and semantics are being used.

[0125]

In this invention, interconnection of the service is carried out with a business system. Therefore, a virtual ENTA prize is formed, and business makes it in between the document (XML coding was carried out) accepted and generated, and is mainly defined as it as follows.

"When an order for your company is placed in purchase, our company accepts it and things are made, a transmittal letter is sent to your company". ["when your company is asked for a catalog by our company a catalog is sent to your company", and]

Explanation of the desirable example of this invention mentioned above is made in order that this invention may explain. This does not mean limiting to a strict form which indicated or [comprehensive] and invention. Many corrections and change are clear for a person skilled in the art. The range of this invention is defined by the following claims and what is equivalent to it.

[Brief Description of the Drawings]

[Drawing 1]

It is a schematic diagram of an electronic commerce network including the business interface definition (BID) by this invention.

[Drawing 2]

It is a schematic diagram of the business interface definition document by this invention.

[Drawing 3]

It is a notional block diagram of the server for the participant node in the network by this invention.

[Drawing 4]

It is a flow chart explaining processing of the document received by the participant node by this invention.

[Drawing 5]

They are a syntax analyzer about the system of an XML base, and a block diagram of a transaction process front end.

[Drawing 6]

It is a notional diagram of the flow of a syntax analyzer function.

[Drawing 7]

It is a schematic diagram of the resource in the server used for the business interface definition by this invention.

[Drawing 8]

It is a schematic diagram of the repository by this invention used for construction of a business interface definition.

[Drawing 9]

It is a flow chart explaining the process of building the business interface definition by this invention.

[Drawing 10]

The heuristic outline of the repository by this invention is provided.

[Drawing 11]

It is a schematic diagram of the resource in the server which provides the market maker function for the network of this invention based on a business interface definition.

[Drawing 12]

It is a flow chart of market maker node processing of the received document.

[Drawing 13]

It is a flow chart explaining the process of a participant's registration in the market maker node by this invention.

[Drawing 14]

It is a flow chart explaining the process of offer of the specifications of the service in the market maker node by the process of drawing 9.

[Drawing 15]

It is a diagram explaining the sequence of the operation in the market maker or participant by this invention.

[Drawing 16]

It is a notional diagram of the element of a commercial transaction network based on BID by this invention.

[Translation done.]

CLAIMS

[Claim(s)]
[Claim 1]The interface (interface) characterized by comprising the following for a transaction between nodes in a network containing two or more nodes (nodes) which perform processing relevant to the transaction (transaction).
. Include translation information which supplies a definition of input document (document), and a definition of an output document. It is the machine-readable specification (machine readable specification) of an interface to transaction processing recorded on an accessible memory by at least one node in a NETTO work, Description of a group of a recorder with an individual definition of said input and an output document.
And the logical structure over a group of said recorder.

[Claim 2]The interface according to claim 1, wherein said translation information contains data type (data type) specification to at least one logical structure in a definition of said input and an output document.
[Claim 3]Said translation information a recorder of a group to the specific logical structure in a definition of said input and an output document decided beforehand, The interface according to claim 1 including at least one data structure which carries out mapping (mapping) in individual entry (entries) of the list (list).
[Claim 4]The interface comprising according to claim 1:
A library of the logical structure.
And a repository [in / by at least one node in a network which records translation information over the logical structure / an accessible memory].

[Claim 5]The interface according to claim 1 containing a document according to a definition of an interface document characterized by comprising the following.

The logical structure for said machine-readable specification to record an identifier of a specific transaction.
And it is at least one of references to a definition and a definition of an input and an output document to said specific transaction.

[Claim 6]In order that said machine-readable specification may record an identifier of an interface, With and said interface. The interface according to claim 1 containing a document according to a definition of an interface document including the logical structure for recording at least one of references to one or more specifications and specifications of a transaction of a lot which are supported.
[Claim 7]Said machine-readable specification including reference to specification of a specific transaction and specification of said specific transaction, The interface according to claim 6 containing a document including the logical structure for recording at least one of references to a definition and a definition of an input and an output document to said specific transaction.
[Claim 8]The interface according to claim 1, wherein said recorder possesses parsed data (parsed).
[Claim 9]Character data in which said parsed data in at least one of said input and the output documents codes the text character (text character) in one of said input and the output documents: (character.) data; it reaches. The interface possessing marking data which identifies a group of a recorder according to the one logical structure of said input and an output document according to claim 8.
[Claim 10]The interface according to claim 9, wherein at least one of the groups of said recorder codes two or more text characters which supply a word of natural language.
[Claim 11]The interface according to claim 8, wherein said translation information over at least one of the groups of said recorder identified by at least one specific logical structure of said input and an output document codes an individual definition to a group of a parsed character.
[Claim 12]The interface according to claim 8, wherein said recorder possesses data which is not parsed.
[Claim 13]In a network of a document type for use in two or more transactions, The interface according to claim 1 characterized by said input and one definition of an output document including reference to a document type in said repository including a repository recorded on an accessible memory by at least one node.
[Claim 14]A method according to claim 13, wherein said repository of a document type contains a document type for identifying participant processing (participant processes) in a network.
[Claim 15]The interface according to claim 1, wherein a definition of said input and an output document possesses a document type definition according to standard extensible marking

language XML.
[Claim 16]The interface according to claim 1, wherein said machine-readable data structure including translation information possesses a document composed according to a document type definition according to standard extensible marking language XML.
[Claim 17]A device comprising:
A network interface.
And according to the transaction processing architecture, It is a device for setting up a participant interface (participant interface) to a transaction performed on a system containing a data-processing resource which performs transaction processing of a transaction. : By said system. It is a program of a command which supplies a tool for performing, being recorded on an accessible medium by said system, and building a definition of a participant interface to a participant in a specific transaction, A definition of an input document [as opposed to said participant in a definition of said participant interface].
And machine-readable description of a group of a recorder with a definition of said input and an output document individual including a definition of an output document to said participant.
And a data structure corresponding to [can perform by said program and; possessing the logical structure over a group of said recorder, and said system, and it is recorded on an accessible medium by said system, and] a group of said recorder, And in order to compile the logical structure of said input and an output document according to said transaction processing architecture, In order to translate into said corresponding data structure, said input document, In order to compile a command which can be executed by said system, and in order to translate an output of said transaction processing into the logical structure of a group of said recorder, and said output document, A program which answers a definition of said input and an output document in order to compile a command which can be executed by said system.

[Claim 18]Said tool for building a definition of a participant interface, In order to access the element (element) of said definition from a repository, including a excecutable command by said system said repository, The device according to claim 17 recording translation information over the logical structure used in order to build a library of the logical structure, and interface description.
[Claim 19]The device according to claim 18, wherein said repository records a definition of a document possessing the logical structure.
[Claim 20]For said access characterized by comprising the following; it reaches. Like a definition of an input document of said interface currently built, The device according to claim 17 including a command which can be executed by said system in order to set up a definition of said participant interface by including a definition of said output document of said complementary transaction.

Said tool for building a definition of a participant interface, A definition of an input document [as opposed to / are for accessing a definition of other participant interfaces to a complementary transaction, and / said complementary transaction in said accessed definition] And a definition of an output document to said complementary transaction.

[Claim 21]Said tool for building a definition of a participant interface, The device according to claim 20 including a command which can be executed by said system like a definition of said output document of an interface currently built since a definition of an input document of said complementary transaction is included.
[Claim 22]The device comprising according to claim 17:
A document according to a definition of a participant interface document including the logical structure for said tool for building a definition of a participant interface to record an identifier of a specific transaction.
And a command which can be executed by said system in order to build at least one of references to a definition and a definition of an input and an output document to said specific transaction.

[Claim 23]Said tool for building a definition of a participant interface, In order to record an identifier of said participant interface, And in order to build a document according to a definition of a participant interface document including the logical structure for recording at least one of references to one or more specifications and specifications of a transaction of a lot which are supported by said participant interface, The device according to claim 17 including a command which can be executed by said system.
[Claim 24]Said document according to a definition of a participant interface document, Including reference to machine-readable specification of a specific transaction, and specification of said specific transaction, The device according to claim 23 containing a document including the logical structure for recording at least one of references to a definition and a definition of an input and an output document to said specific transaction.
[Claim 25]The device according to claim 17, wherein said recorder possesses parsed data.
[Claim 26]Character data in which said parsed data in at least one of said input and the output documents codes the text character (text character) in one of said input and the output documents: (character.) data; it reaches. The device possessing marking data which identifies a group of a recorder according to the one logical structure of said input and an output document according to claim 25.
[Claim 27]The device according to claim 26, wherein at least one of the groups of said recorder codes two or more text characters which supply a word of natural language.
[Claim 28]Said specification includes translation information over at least one of groups of said

recorder identified by said at least one logical structure among said input and an output document, The device according to claim 25 coding an individual definition to a group of a parsed character.
[Claim 29]The device according to claim 25, wherein said recorder possesses data which is not parsed.
[Claim 30]The device according to claim 17, wherein the logical structure of a data structure corresponding to a group of said recorder, said input, and an output document contains a programming object containing a variable and a method of having followed the variable transaction processing architecture.
[Claim 31]The device according to claim 17, wherein said variable transaction processing architecture of said transaction processing possesses processing according to interface description language.
[Claim 32]The device according to claim 17, wherein a definition of said input and an output document possesses a document type definition according to standard extensible marking language XML.
[Claim 33]The device according to claim 19, wherein said input and said one definition of an output document include reference to a document type in a repository.
[Claim 34]The device according to claim 18, wherein said repository includes translation information which specifies measurement of a product subject of a transaction.
[Claim 35]The device according to claim 18, wherein said repository includes translation information which specifies the cost (cost) of a product subject of a transaction.
[Claim 36]The device according to claim 18, wherein said repository includes translation information which specifies the characteristic of a product subject of a transaction.
[Claim 37]The device according to claim 18, wherein said repository includes translation information which specifies a financial term (financialterms) of a transaction.
[Claim 38]The device according to claim 18, wherein said repository includes translation information which specifies a term of shipment (shipment) to a product subject of a transaction.

[Claim 39]It is a device for setting up a participant interface to a transaction performed on a system. : A memory which records data and a program of a command;
It is the data processor characterized by comprising the following connected to a memory which executes said program of a command, and is a program of the; aforementioned command, A tool for building a definition of a participant interface for a participant in a specific transaction
Machine-readable description of a group of a recorder with a definition of said input and an output document individual including a definition of an input document [as opposed to said participant in said definition of a participant interface], and a definition of an output document

to said participant.
And the logical structure over a group of said recorder.

[Claim 40]Said tool for building a definition of a participant interface including a repository recorded on an accessible memory by said data processor, In order to access an element of said definition from a repository, including a excecutable command by said system said repository, The device according to claim 39 recording translation information over the logical structure used in order to build a library of the logical structure, and interface description.
[Claim 41]The device according to claim 40, wherein said repository records a definition of a document possessing the logical structure.
[Claim 42]For said access characterized by comprising the following; it reaches. Like a definition of an input document of said interface currently built, The device according to claim 39 including a command which can be executed by said system in order to set up a definition of said participant interface by including a definition of said output document of said complementary transaction.
Said tool for building a definition of a participant interface, A definition of an input document [as opposed to / are for accessing a definition of other participant interfaces to a complementary transaction, and / said complementary transaction in said accessed definition] And a definition of an output document to said complementary transaction.

[Claim 43]Said tool for building a definition of a participant interface, The device according to claim 42 including a command which can be executed by said system like a definition of said output document of an interface currently built since a definition of an input document of said complementary transaction is included.
[Claim 44]The device comprising according to claim 39:
A document according to a definition of a participant interface document including the logical structure for said tool for building a definition of a participant interface to record an identifier of a specific transaction.
And a command which can be executed by said system in order to build at least one of references to a definition and a definition of an input and an output document to said specific transaction.

[Claim 45]Said tool for building a definition of a participant interface, In order to record an identifier of said participant interface, And in order to build a document according to a definition of a participant interface document including the logical structure for recording at least one of references to one or more specifications and specifications of a transaction of a lot which are supported by said participant interface, The device according to claim 39 including a command

which can be executed by said system.
[Claim 46]Said document according to a definition of a participant interface document, Including reference to machine-readable specification of a specific transaction, and specification of said specific transaction, The device according to claim 45 containing a document including the logical structure for recording at least one of references to a definition and a definition of an input and an output document to said specific transaction.
[Claim 47]The device according to claim 25, wherein said recorder possesses parsed data.
[Claim 48]said parsed data in at least one of said input and the output documents: character data which codes a text character in one of said input and the output documents, and; -- and -- A group of a recorder, The device possessing marking data identified according to the one logical structure of said input and an output document according to claim 47.
[Claim 49]The device according to claim 48, wherein at least one of the groups of said recorder codes two or more text characters which supply a word of natural language.
[Claim 50]Said specification includes translation information over at least one of groups of said recorder identified by said at least one logical structure among said input and an output document, The device according to claim 47 coding an individual definition to a group of a parsed character.
[Claim 51]The device according to claim 47, wherein said recorder possesses data which is not parsed.
[Claim 52]The device according to claim 39, wherein the logical structure of a data structure corresponding to a group of said recorder, said input, and an output document contains a programming object containing a variable and a method of having followed the variable transaction processing architecture.
[Claim 53]The device according to claim 39, wherein said variable transaction processing architecture of said transaction processing possesses processing according to interface description language.
[Claim 54]The device according to claim 39, wherein a definition of said input and an output document possesses a document type definition according to standard extensible marking language XML.
[Claim 55]The device according to claim 41, wherein said input and said one definition of an output document include reference to a document type in a repository.
[Claim 56]The device according to claim 40, wherein said repository includes translation information which specifies measurement of a product subject of a transaction.
[Claim 57]The device according to claim 40, wherein said repository includes translation information which specifies cost of a product subject of a transaction.
[Claim 58]The device according to claim 40, wherein said repository includes translation information which specifies the characteristic of a product subject of a transaction.

[Claim 59]The device according to claim 40, wherein said repository includes translation information which specifies a financial term of a transaction.
[Claim 60]The device according to claim 40, wherein said repository includes translation information which specifies a term of shipment to a product subject of a transaction.
[Claim 61]A method comprising:
It is a method for programming a commercial transaction in a network. : A machine-readable definition of an input document to a node in a network containing a resource for performing processing in said transaction
And are a stage to define a machine-readable definition of an output document to said node, and a definition of said input and an output document, An individual description of a group of a recorder, and said stage of providing the logical structure over a group of said recorder; it reaches. A stage which supplies translation information over said logical structure to said node

[Claim 62]A method according to claim 61, wherein said translation information contains data type specification to at least one logical structure in a definition of said input and an output document.
[Claim 63]Said translation information a recorder of a group to the specific logical structure in a definition of said input and an output document decided beforehand, A method according to claim 61, wherein the claim aforementioned translation information including at least one data structure mapped for an individual entry of a list contains specification of a data type to at least one logical structure in a definition of said input and an output document.
[Claim 64]A method comprising according to claim 61:
Said stage which supplies translation information is a library of the logical structure.
And a stage which supplies a repository in an accessible memory by at least one node in a network which records translation information over the logical structure.

[Claim 65]A method according to claim 61 including a stage of defining machine-readable specification of an interface containing a document according to a definition of an interface document characterized by comprising the following.
The logical structure for recording an identifier of a specific transaction.
And the logical structure for recording at least one of references to a definition and a definition of an input and an output document to said specific transaction.

[Claim 66]A method according to claim 61, wherein said recorder possesses parsed data.
[Claim 67]Said parsed data in at least one of said input and the output documents: A group of character data which codes a text character in one of said input and the output documents,;, and a recorder, A method possessing marking data identified according to the one logical

structure of said input and an output document according to claim 66.
[Claim 68]A method according to claim 67, wherein at least one of the groups of said recorder codes two or more text characters which supply a word of natural language.
[Claim 69]A method according to claim 67, wherein said translation information over at least one of the groups of said recorder identified by at least one specific logical structure of said input and an output document codes an individual definition to a group of a parsed character.
[Claim 70]A method according to claim 66, wherein said recorder possesses data which is not parsed.
[Claim 71]A method according to claim 67, wherein at least one of the recorders of said group codes two or more text characters which supply a word of natural language.
[Claim 69]A method according to claim 67, wherein said translation information over at least one of the recorders of said group identified by at least one specific logical structure of said input and an output document codes an individual definition to a group of a parsed character.
[Claim 70]A method according to claim 66, wherein said recorder possesses data which is not parsed.
[Claim 71]A method according to claim 61, wherein a definition of said input and an output document possesses a document type definition according to standard extensible marking language XML.
[Claim 72]a stage which supplies the purser (parser) for answering the logical structure and generating an event signal (event signal) in a definition of said input document, and; -- and -- In order to perform said processing, A method according to claim 61 including a stage which supplies an event listener program (event listener programs) which answers said event signal.
[Claim 73]It is a method for performing a transaction between nodes in a network containing two or more nodes which perform processing relevant to a transaction. : It is a stage which records machine-readable specification of an interface to a transaction, Said stage where, as for said specification, a definition of said input and an output document possesses the logical structure over a group of an individual description of a group of a recorder, and said recorder including a definition of an input document, and a definition of an output document;
A stage of receiving data which possesses a document through a communication network;
A stage of parsing said document according to said specification for identifying an input document;
A stage which supplies said at least a part of input document in a machine readable form to transaction processing which generates an output;
A stage which forms an output document which possesses said output according to said definition of an output document based on said specification; it reaches. A method possessing a stage of transmitting said output document on said communication network.
[Claim 74]A stage which accesses specification of a complementary interface supplied to other

nodes in a network to said transaction, comprising:
A definition of an input document [as opposed to said complementary interface in said accessed specification].
And a definition of an output document to said complementary interface.

[Claim 75]A method according to claim 74 including a stage of finding said complementary interface in a network.
[Claim 76]A stage where said stage characterized by comprising the following of setting up said recorded specification accesses an element of machine-readable specification from a repository.
Said repository is a library of the logical structure.
A mimetic diagram map to the logical structure (schematic map).
And the logical structure used in order to build interface description.

[Claim 77]By including said at least a part of definition of said input document of said complementary interface in said definition of said output document of an interface in said recorded specification, A method according to claim 74 including a stage of setting up said recorded specification of said interface.
[Claim 78]A method according to claim 73 including a stage which supplies access to said specification which let a communication network pass to other nodes in a NETTO work.
[Claim 79]A method according to claim 73, wherein access to said specification includes said stage supplied to other nodes in a network in said network including a stage which transmits specification of said interface to other nodes in a network.
[Claim 80]A method according to claim 73 containing a document according to a definition of an interface document characterized by comprising the following.
The logical structure for said machine-readable specification to record an identifier of a specific transaction.
And it is at least one of references to a definition and a definition of an input and an output document to said specific transaction.

[Claim 81]In order that said machine-readable specification may record an identifier of an interface, With and said interface. A method according to claim 73 containing a document according to a definition of an interface document including the logical structure for recording at least one of references to one or more specifications and specifications of a transaction of a lot which are supported.
[Claim 82]Said machine-readable specification including reference to specification of a specific transaction and specification of said specific transaction, A method according to claim 81

containing a document including the logical structure for recording at least one of references to a definition and a definition of an input and an output document to said specific transaction.
[Claim 83]A method according to claim 73, wherein said recorder possesses parsed data (parsed).
[Claim 84]said parsed data in at least one of said input and the output documents: character data which codes a text character in one of said input and the output documents, and; -- and -- A group of a recorder, A method possessing marking data identified according to the one logical structure of said input and an output document according to claim 83.
[Claim 85]A method according to claim 84, wherein at least one of the groups of said recorder codes two or more text characters which supply a word of natural language.
[Claim 86]Said specification includes said translation information over at least one of groups of said recorder identified by at least one logical structure among said input and an output document, A method of coding an individual definition to a group of a parsed character according to claim 83.
[Claim 87]A method according to claim 83, wherein said recorder possesses data which is not parsed.
[Claim 88]Said transaction processing has the variable transaction processing architecture, And a method according to claim 73 including a stage of translating said at least a part of input document into form which can be read according to said variable transaction processing architecture of said transaction processing.
[Claim 89]A method according to claim 88, wherein said translate phase includes a stage which generates a programming object containing a variable and a method of having followed said variable transaction processing architecture of said transaction processing.
[Claim 90]A method according to claim 88, wherein said variable transaction processing architecture of said transaction processing possesses processing according to interface description language.
[Claim 91]A method according to claim 73, wherein a definition of said input and an output document possesses a document type definition according to standard extensible marking language XML.
[Claim 92]A method according to claim 73 characterized by said input and said one definition of an output document including reference to a document type in said repository including a stage which supplies a repository of a document type for use in two or more transactions.
[Claim 93]A method according to claim 92, wherein said repository of a document type contains a document type for identifying participant processing in a network.
[Claim 94]A method of including translation information which identifies a parameter of a transaction including a stage which supplies a repository of translation information to the logical structure according to claim 92.

[Claim 95]Said transaction processing has the variable transaction processing architecture, It reaches. : A definition of an input document [as opposed to / are a stage which accesses specification of a complementary interface and / said complementary interface, in said accessed specification], and said stage of including a definition of an output document to said complementary interface;
Like a definition of said input document of an interface in recorded specification, By including a definition of said output document of said complementary interface, A stage of setting up specification on which an interface was recorded; it reaches. A definition of said input and an output document is answered, The logical structure of said input and an output document according to a data structure corresponding to a group of a recorder, and said transaction processing architecture of said transaction processing, In order to translate said input document into said corresponding data structure, A method of including a stage which compiles a command which can be executed by said system, in order to translate into form according to a definition of said output document an output of a command which can be executed by said system, and said transaction processing according to claim 73.
[Claim 96]A method according to claim 95 including a stage of setting up specification on which an interface was recorded by including a definition of said input document of said complementary interface like a definition of said output document of an interface in recorded specification.
[Claim 97]A definition of a document characterized by comprising the following is recorded, and it reaches.;
A stage of said transaction processing having the variable transaction processing architecture, and setting up said recorded specification includes a stage which accesses an element of said machine-readable specification from a repository, and said repository is a library of the logical structure.
A mimetic diagram map to the logical structure.
And the logical structure used in order to build interface description.

The logical structure of said input and an output document which answered a definition of said input and an output document, and followed a data structure corresponding to a group of a recorder, and said transaction processing architecture of said transaction processing, In order to translate said input document into said corresponding data structure, A method of including a stage which compiles a command which can be executed by said system, in order to translate into form according to a definition of said output document an output of a command which can be executed by said system, and said transaction processing according to claim 73.
[Claim 98]It is a device for managing a transaction between nodes in a network containing two or more nodes which perform processing relevant to a transaction. : A network interface;

Are a memory which records data and a program of a command, and machine-readable specification of an interface to a transaction is included, Said memory for which, as for said specification, a definition of said input and an output document possesses the logical structure over a group of an individual description of a group of a recorder, and said recorder including a definition of a definition of an input document, and an output document;
Data which is the data processor connected to said network interface which executes a program of said memory and a command, and possesses a document through a program of the; aforementioned command, and a network interface Logic for receiving;
Said specification for identifying an input document is followed, and it is said document. Logic for parsing;
An output is generated for said at least a part of input document in a machine readable form. Logic for supplying transaction processing to carry out;
Based on said specification, according to a definition of an output document, It is ****** about said output. Logic for forming a ****** document; it reaches. In order to transmit said output document on a network interface A device possessing said data processor including logic.
[Claim 99]It is the logic for accessing specification of a complementary interface, Said accessed specification to said complementary interface. A definition of a receiving input document, and said logic including a definition of an output document to said complementary interface; it reaches. Like a definition of said input document of said interface in recorded specification, The device according to claim 98 including logic for setting up said recorded specification of said interface by including a definition of said output document of said complementary interface.
[Claim 100]The device according to claim 99 recording a definition of a document characterized by comprising the following.
Said logic for setting up said recorded specification includes logic for accessing an element of said machine-readable specification from a repository, and said repository is a library of the logical structure.
A mimetic diagram map to the logical structure.
And the logical structure used in order to build interface description.

[Claim 101]Like a definition of said output document of said interface in recorded specification, The device according to claim 99 including logic for setting up said recorded specification of said interface by including a definition of said input document of said complementary interface.
[Claim 102]The device according to claim 98 including logic which supplies access to said specification which let a communication network pass to other nodes in a NETTO work.
[Claim 103]The device according to claim 98, wherein access to said specification includes said logic supplied to other nodes in a network in said network including logic for transmitting

specification of said interface to other nodes in a network.
[Claim 104]The device according to claim 98 containing a document according to a definition of an interface document characterized by comprising the following.
The logical structure for said machine-readable specification to record an identifier of a specific transaction.
And it is at least one of references to a definition and a definition of an input and an output document to said specific transaction.

[Claim 105]In order that said machine-readable specification may record an identifier of an interface, With and said interface. The device according to claim 98 containing a document according to a definition of an interface document including the logical structure for recording at least one of references to one or more specifications and specifications of a transaction of a lot which are supported.
[Claim 106]Said machine-readable specification including reference to specification of a specific transaction and specification of said specific transaction, The device according to claim 105 containing a document including the logical structure for recording at least one of references to a definition and a definition of an input and an output document to said specific transaction.
[Claim 107]The device according to claim 98, wherein said recorder possesses parsed data.
[Claim 108]said parsed data in at least one of said input and the output documents: character data which codes a text character in one of said input and the output documents, and; -- and -- A group of a recorder, The device possessing marking data identified according to the one logical structure of said input and an output document according to claim 107.
[Claim 109]The device according to claim 108, wherein at least one of the groups of said recorder codes two or more text characters which supply a word of natural language.
[Claim 110]Said specification includes said translation information over at least one of groups of said recorder identified by at least one logical structure among said input and an output document, The device according to claim 107 coding an individual definition to a group of a parsed character.
[Claim 111]The device according to claim 107, wherein said recorder possesses data which is not parsed.
[Claim 112]Said transaction processing has the variable transaction processing architecture, And the device according to claim 98 including logic which translates said at least a part of input document into form which can be read according to said variable transaction processing architecture of said transaction processing.
[Claim 113]The device according to claim 112, wherein logic for [said] translating includes logic which generates a programming object containing a variable and a method of having

followed said variable transaction processing architecture of said transaction processing.
[Claim 114]The device according to claim 112, wherein said variable transaction processing architecture of said transaction processing includes processing according to interface description language.
[Claim 115]The device according to claim 98, wherein a definition of said input and an output document possesses a document type definition according to standard extensible marking language XML.
[Claim 116]An accessible memory is included by a processor which records a repository of a document type for use in two or more transactions, And the device according to claim 98, wherein said input and said one definition of an output document include reference to a document type in said repository.
[Claim 117]The device according to claim 116, wherein said repository of a document type contains a document type for identifying participant processing in a network.
[Claim 118]The device according to claim 116 including translation information which identifies a parameter of a transaction including a stage which supplies a repository of translation information to the logical structure.
[Claim 119]Said transaction processing has the variable transaction processing architecture, It reaches. : A definition of an input document [as opposed to / are the logic for accessing specification of a complementary interface, and / said complementary interface in said accessed specification], and said logic including a definition of an output document to said complementary interface;
The device comprising according to claim 98:
Like a definition of said input document of an interface in recorded specification, By including a definition of said output document of said complementary interface, Logic which sets up specification on which an interface was recorded; it reaches. A definition of said input and an output document is answered, In order to compile the logical structure of said input and an output document according to a data structure corresponding to a group of a recorder, and said transaction processing architecture of said transaction processing.
In order to compile a command which can be executed by said system in order to translate said input document into said corresponding data structure.
And logic for compiling a command which can be executed by said system, in order to translate an output of said transaction processing into form according to a definition of said output document.

[Claim 120]The device according to claim 119 including logic for setting up specification on which an interface was recorded by including a definition of said input document of said complementary interface like a definition of said output document of an interface in recorded

specification.
[Claim 121]A definition of a document characterized by comprising the following is recorded, and it reaches.;
By said transaction processing's having the variable transaction processing architecture, and accessing an element of said machine-readable specification from a repository, logic for setting up said recorded specification is included, and said repository is a library of the logical structure.
A mimetic diagram map to the logical structure.
And the logical structure used in order to build interface description.

The device comprising according to claim 98:
In order to compile the logical structure of said input and an output document which answered a definition of said input and an output document, and followed a data structure corresponding to a group of a recorder, and said transaction processing architecture of said transaction processing.
In order to compile a command which can be executed by said system in order to translate said input document into said corresponding data structure.
And logic which compiles a command which can be executed by said system in order to translate an output of said transaction processing into form according to a definition of said output document.

[Claim 122]It is a device for managing a transaction between nodes in a network containing two or more nodes which perform processing relevant to a transaction. : It is a stage which records machine-readable specification of two or more participant interfaces, Said participant interface identifies a transaction and said individual transaction is identified by a definition of an input document, and definition of an output document, Said stage where said definition of said input and an output document possesses the logical structure over a group of an individual description of a group of a recorder, and said recorder;
A stage of receiving data which possesses a document through a communication network;
Said specification for identifying an input document, and a stage of parsing said document according to one or more transactions which receive said identified input document;
A method possessing a stage which supplies said at least a part of input document in a machine readable form to transaction processing relevant to said one or more identified transactions.
[Claim 123]A method according to claim 122 including a stage which supplies a repository which records a definition of a document characterized by comprising the following.
A library of the logical structure.

A mimetic diagram map to the logical structure.
And the logical structure used in order to build participant interface description.

[Claim 124]A method according to claim 123 including a stage which supplies access to said repository which let a communication network pass to other nodes in a NETTO work.
[Claim 125]A method according to claim 122 containing a document according to a definition of a participant interface document characterized by comprising the following.
The logical structure for said machine-readable specification to record an identifier of a specific transaction.
And it is at least one of references to a definition and a definition of an input and an output document to said specific transaction.

[Claim 126]In order that said machine-readable specification may record an identifier of a participant interface, With and said participant interface. A method according to claim 122 containing a document according to a definition of an interface document including the logical structure for recording at least one of references to one or more specifications and specifications of a transaction of a lot which are supported.
[Claim 127]Said document according to a definition of a participant interface document, Including reference to specification of a specific transaction, and specification of said specific transaction, A method according to claim 126 containing a document including the logical structure for recording at least one of references to a definition and a definition of an input and an output document to said specific transaction.
[Claim 128]A method according to claim 122, wherein said recorder possesses parsed data.
[Claim 129]Said parsed data in at least one of said input and the output documents: A group of character data which codes a text character in one of said input and the output documents,;, and a recorder, A method possessing marking data identified according to the one logical structure of said input and an output document according to claim 128.
[Claim 130]A method according to claim 109, wherein at least one of the groups of said recorder codes two or more text characters which supply a word of natural language.
[Claim 131]Said specification includes said translation information over at least one of groups of said recorder identified by at least one logical structure among said input and an output document, A method of coding an individual definition to a group of a parsed character according to claim 130.
[Claim 132]A method according to claim 130, wherein said recorder possesses data which is not parsed.
[Claim 133]A method comprising according to claim 122:
A stage which supplies said at least a part of input document in a machine readable form to

transaction processing relevant to one or more identified transactions, It reaches including a stage of performing routing processing according to the processing architecture. : A definition of said input in said participant interface and an output document is answered, The logical structure of said input and an output document according to a data structure corresponding to said group of a recorder, and the processing architecture of said transaction processing.
And a stage which compiles a command which can be executed by said system in order to translate said input document into said corresponding data structure.

[Claim 134]A method comprising according to claim 122:
A stage where a stage which supplies said at least a part of input document in a machine readable form to transaction processing relevant to one or more identified transactions performs routing processing according to the processing architecture.
And a stage of translating said at least a part of input document into form which can be read according to said processing architecture.

[Claim 135]A method according to claim 134, wherein said translate phase includes a stage which generates a programming object containing a variable and a method of having followed said processing architecture of said routing processing.
[Claim 136]A stage which supplies said at least a part of input document in a machine readable form to transaction processing relevant to one or more identified transactions, A method according to claim 122 including a stage which carries out routing of said portion of said input document to said identified transaction.
[Claim 137]A method according to claim 136, wherein said routing stage includes a stage which transmits said input document on a communication network to a node which performs one of said the identified transactions.
[Claim 138]A method according to claim 122, wherein a definition of said input and an output document possesses a document type definition according to standard extensible marking language XML.
[Claim 139]A method according to claim 138, wherein specification of a participant interface possesses a definition of a document according to a document type definition according to standard extensible marking language XML.
[Claim 140]Said repository including a document type with which it was standardized for use in two or more transactions and said input and said one definition of an output document, A method according to claim 122 including reference to a standardized document type in a repository.
[Claim 141]A method according to claim 140, wherein said repository contains a document type with which it was standardized for identifying participant processing in a network.

[Claim 142]A method of including translation information which identifies a parameter of a transaction including a stage which supplies a repository of translation information to the logical structure according to claim 140.
[Claim 143]A method comprising according to claim 122:
Said transaction processing has respectively one of two or more of the variable transaction processing architecture, And a stage of translating said at least a part of input document into form which can be read according to said variable transaction processing architecture of said individual transaction processing.
And a stage which carries out routing of said translated portion to said individual transaction processing.

[Claim 144]A method according to claim 143, wherein said translate phase includes a stage which generates a programming object which contains a variable and a method according to the variable transaction processing architecture of said individual transaction processing.
[Claim 145]A method according to claim 144, wherein said variable transaction processing architecture of said transaction processing possesses processing according to interface description language.
[Claim 146]It is a device for managing a transaction between nodes in a network containing two or more nodes which perform processing relevant to a transaction. : A network interface; Are a memory which records data and a program of a command, and machine-readable specification of two or more participant interfaces is included, Identify said participant interface and a transaction said individual transaction, Said memory in which a definition of said input and an output document possesses the logical structure over a group of an individual description of a group of a recorder, and said recorder by being identified by a definition of an input document, and definition of an output document;
Data which is the data processor connected to said network interface which executes a program of said memory and a command, and possesses a document through a program of the; aforementioned command, and a network interface Logic for receiving;
Specification and an identified input document for identifying an input document According to one or more transactions to receive, It is pass ** about said document. Said at least a part of input document in logic of a ** sake,;, and a machine readable form. One or more A device possessing said data processor including logic for supplying transaction processing relevant to an identified transaction.
[Claim 147]The device according to claim 146 containing a repository recorded on an accessible memory by said data processor which records a definition of a document characterized by comprising the following.
A library of the logical structure.

The logical structure used in order to build a mimetic diagram map and participant interface description to the logical structure.

[Claim 148]The device according to claim 146 including logic characterized by comprising the following for accessing to a repository recorded on a memory through a network interface which records a definition of a document.
A library of the logical structure.
The logical structure used in order to build a mimetic diagram map and participant interface description to the logical structure.

[Claim 149]The device according to claim 146 containing a document according to a definition of a participant interface document characterized by comprising the following.
The logical structure for said machine-readable specification to record an identifier of a specific transaction.
And it is at least one of references to a definition and a definition of an input and an output document to said specific transaction.

[Claim 150]In order that said machine-readable specification may record an identifier of a participant interface, With and said participant interface. The device according to claim 146 containing a document according to a definition of an interface document including the logical structure for recording at least one of references to one or more specifications and specifications of a transaction of a lot which are supported.
[Claim 151]Said document according to a definition of a participant interface document, Including reference to specification of a specific transaction, and specification of said specific transaction, The device according to claim 150 containing a document including the logical structure for recording at least one of references to a definition and a definition of an input and an output document to said specific transaction.
[Claim 152]The device according to claim 146, wherein said recorder possesses parsed data.
[Claim 153]said parsed data in at least one of said input and the output documents: character data which codes a text character in one of said input and the output documents, and; -- and -- A group of a recorder, The device possessing marking data identified according to the one logical structure of said input and an output document according to claim 152.
[Claim 154]The device according to claim 153, wherein at least one of the groups of said recorder codes two or more text characters which supply a word of natural language.
[Claim 155]Said specification includes said translation information over at least one of groups of said recorder identified by at least one logical structure among said input and an output document, The device according to claim 154 coding an individual definition to a group of a

parsed character.
[Claim 156]The device according to claim 154, wherein said recorder possesses data which is not parsed.
[Claim 157]The device comprising according to claim 146:
Logic for supplying said at least a part of input document in a machine readable form to transaction processing relevant to one or more identified transactions, It reaches including routing processing according to the processing architecture. : A definition of said input in said participant interface and an output document is answered, In order to compile the logical structure of said input and an output document according to a data structure corresponding to said group of a recorder, and the processing architecture of said transaction processing.
And a compiler for compiling a command which can be executed by said system, in order to translate said input document into said corresponding data structure.

[Claim 158]Logic for supplying said at least a part of input document in a machine readable form to transaction processing relevant to one or more identified transactions, The device according to claim 146 including logic for translating at least a part of :aforementioned input document into form which can be read according to said processing architecture including routing processing according to the processing architecture.
[Claim 159]The device according to claim 158, wherein logic for [said] translating includes a stage which generates a programming object containing a variable and a method of having followed the processing architecture of said routing processing.
[Claim 160]Logic for supplying said at least a part of input document in a machine readable form to transaction processing relevant to one or more identified transactions, The device according to claim 146 containing the router (router) for carrying out routing of said portion of said input document to said identified transaction.
[Claim 161]The device according to claim 160, wherein said router includes logic for transmitting said input document on a network interface to a node which performs one of said the identified transactions.
[Claim 162]The device according to claim 146, wherein a definition of said input and an output document possesses a document type definition according to standard extensible marking language XML.
[Claim 163]The device according to claim 162, wherein specification of a participant interface possesses a definition of a document according to a document type definition according to standard extensible marking language XML.
[Claim 164]Said repository including a document type with which it was standardized for use in two or more transactions and said input and said one definition of an output document, The device according to claim 147 including reference to a standardized document type in a

repository.
[Claim 165]The device according to claim 147, wherein said repository contains a document type with which it was standardized for identifying participant processing in a network.
[Claim 166]Said transaction processing has respectively one of two or more of the variable transaction processing architecture, And in order to translate said at least a part of input document into form which can be read according to said variable transaction processing architecture of said individual transaction processing, And the device according to claim 146 including logic for carrying out routing of said translated portion to said individual transaction processing.
[Claim 167]The device according to claim 166, wherein logic for [said] translating generates a programming object which contains a variable and a method according to said variable transaction processing architecture of said individual transaction processing.
[Claim 168]The device according to claim 166, wherein said variable transaction processing architecture of said transaction processing possesses processing according to interface description language.

[Translation done.]